Protein Superfamily Classification using Computational Intelligence Techniques

Ph.D. THESIS

by

SWATI VIPSITA

Department of Computer Science and Engineering

National Institute of Technology Rourkela

Rourkela- 769008, India

JANUARY 2014

Protein Superfamily Classification using Computational Intelligence Techniques

Dissertation submitted to the department of

Computer Science and Engineering

of

National Institute of Technology Rourkela

in partial fulfilment of the requirements

for the degree of

Doctor of Philosophy

by

Swati Vipsita

(Roll- 509CS101)

under the supervision of

Prof. Santanu Kumar Rath

Department of Computer Science and Engineering

National Institute of Technology Rourkela

Rourkela- 769008, India

2014

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, India. www.nitrkl.ac.in

Dr. Santanu K. Rath
Professor

January 29, 2014

Certificate

This is to certify that the work in the thesis entitled ***Protein Superfamily Classification using Computational Intelligence Techniques*** by ***Swati Vipsita*** is a record of an original research work carried out by her under my supervision and guidance in partial fulfilment of the requirements for the award of the degree of *Doctor of Philosophy* in *Computer Science and Engineering*. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

(Santanu K. Rath)

Acknowledgment

The famous English poet in Stratford-upon-Avon said "Pupil Thy Work is Incomplete, till thee thank the Lord and thy Master", which means a students work is incomplete until he thanks the Almighty and his Teacher. I sincerely believe in this. I sincerely thank God for showing me the right direction.

I would like to express my indebted thanks to my supervisor Prof. S. K. Rath, for his invaluable guidance, and encouragement during the course of this thesis. His keen interest, patient hearing and constructive criticism have instilled in me the spirit of confidence to successfully complete this thesis. I am greatly indebted for his help throughout the thesis work.

I am very much indebted to the Doctoral Scrutiny committee members Prof. S. K. Jena, Prof. A. K. Turuk, Prof. P. M. Khilar and Prof. G. K. Panda for their time to provide more insightful opinions into my research. Besides that, I am also thankful to all the Professors and faculty members of the department for their in time support, advise and encouragement.

I am really thankful to all my fellow research colleagues for their cooperation. My sincere thanks to Bithin, Suresh, Shashank, Asish, Soubhagya, Pranati, Purabi, Ruby, Reene, Anita for their all support and help. I am truly indebted.

I do acknowledge the academic resources that I have received from NIT Rourkela. I also thank the administrative and technical staff members of the Computer Science Department for their in-time support.

Last, but not the least, I would like to dedicate this thesis to my father for his constant support, encouragement and cooperation. My special thanks to my sister Prachi and brother Satyam for their moral support and cooperation.

Swati Vipsita

Abstract

The problem of protein superfamily classification is a challenging research area in Bioinformatics and has its major application in drug discovery. If a newly discovered protein which is responsible for the cause of new disease gets correctly classified to its superfamily, then the task of the drug analyst becomes much easier. The analyst can perform molecular docking to find the correct relative orientation of ligand for the protein. The ligand database can be searched for all possible orientations and conformations of the protein belonging to that superfamily paired with the ligand. Thus, the search space is reduced enormously as the protein-ligand pair is searched for a particular protein superfamily. Therefore, correct classification of proteins becomes a very challenging task as it guides the analysts to discover appropriate drugs. In this thesis, Neural Networks (NN), Multiobjective Genetic Algorithm (MOGA), and Support Vector Machine (SVM) are applied to perform the classification task.

Adaptive MultiObjective Genetic Algorithm (AMOGA), which is a variation of MOGA is implemented for the structure optimization of Radial Basis Function Network (RBFN). The modification to MOGA is done based on the two key controlling parameters such as probability of crossover and probability of mutation. These values are adaptively varied based upon the performance of the algorithm, i.e., based upon the percentage of the total population present in the best non-domination level. The problem of finding the number of hidden centers remains a critical issue for the design of RBFN. The most optimal RBF network with good generalization ability can be derived from the pareto optimal set. Therefore, every solution of the pareto optimal set gives information regarding the specific samples to be chosen as hidden centers as well as the update weight matrix connecting the hidden and output layer. Principal Component Analysis (PCA) has been used for dimension reduction and significant feature extraction from long feature vector of amino acid sequences.

In two-stage approach for protein superfamily classification, feature extraction process is carried in the first stage and design of the classifier has been proposed 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 eigen vectors from the PCA space which are encoded as binary strings in the chromosome. Using PCA-NSGA-II (non-dominated sorting GA), the non-dominated solutions obtained from the pareto front solves the trade-off problem by compromising between the number of eigen vectors selected and the accuracy obtained by the classifier. In the second stage, Recursive Orthogonal Least Square Algorithm (ROLSA) is used for training RBFN. ROLSA selects the optimal number of

hidden centres as well as updates the output layer weighting matrix. This approach can be applied to large data set with much lower requirements of computer memory. Thus, very small architecture having few numbers of hidden centres are obtained showing higher level of performance accuracy.

As neural networks suffer from two major drawbacks such as getting stuck in local minima and over fitting, so Support vector machine (SVM) is then applied for classification. MOGA selects the optimal number of significant eigen vectors from the eigen space as well as optimize the hyper-parameters of SVM. Using GA based wrapper approach for feature subset selection; the eigen vectors and hyper-parameters of SVM are encoded in the chromosome. SVM classifier is wrapped with every chromosome for evaluating the fitness value. Using MOGA-SVM, the non-dominated solutions obtained from the pareto front solves the trade-off problem by compromising between the number of eigen vectors selected and the accuracy obtained by the classifier. Thus, MOGA-SVM finds a solution between two conflicting objectives of SVM such as model complexity and accuracy. To fasten the convergence process, AMOGA-SVM was implemented and a comparison between MOGA-SVM and AMOGA-SVM was studied.

Each of the proposed work is evaluated separately and their performances are analysed in terms of sensitivity, specificity and accuracy and compared with existing techniques.

Keywords: MOGA, pareto front, non-domination level, probabilities of crossover and mutation, n-gram feature extraction, orthogonal least square algorithm, eigen vector, kernel function, hyper-parameters.

Contents

List of Acronyms/ Abbreviations

A	Alanine
AGA	Adaptive Genetic Algorithm
AMOGA	Adaptive Multiobjective Genetic Algorithm
AVURPSO	Adaptive Velocity Update Relaxation PSO
BLAST	Basic Local Alignment Search Tool
BLOSUM	Blocks Substitution Matrix
BNN	Bayesian Neural Network
BP	Backpropagation
C	Cytosine
CD	Crowding Distance
CATH	Class- Architecture- Topology- Homologous Superfamily
CC	Correlation Coefficient
COGS	Clusters of Orthologous Groups
DE	Differential Evolution
DNA	Deoxyribonucleic Acid
EA	Evolutionary Algorithm
ELM	Extreme Learning Machine
EP	Evolutionary Programming
FAR	False Acceptance Rate
FASTA	FAST-All
FFNN	Feedforward Neural Network
FL	Fuzzy Logic
FN	Number of False Negative
FP	Number of False Positive
FPE	Final Prediction Error
FRR	False Rejection Rate
FSSP	Families of Structurally Similar Proteins
Fuzzy ARTMAP	Fuzzy logic and Adaptive Resonance Theory
G	Guanine

GA	Genetic Algorithm
GNBR	Gauss-Newton Bayesian Regularization
GP	Genetic Programming
GRBF	Generalized Radial Basis Function
HLA	Hybrid Learning Algorithm
HRDGA	Hierarchical Rank Density Genetic Algorithm
HMM	Hidden Markov Model
ICA	Independent Component Analysis
KPCA	Kernel Principal Component Analysis
LDA	Linear Discriminant Analysis
LS	Local Similarity
MDS	Multidimensional Scaling
MSE	Mean Square Error
MIMO	Multi-Input Multi-Output
MOEA	Multiobjective Evolutionary Algorithm
MOGA	Multiobjective Genetic Algorithm
MOP	Multiobjective Optimization Problem
MOPSO	Multiobjective Particle Swarm Optimization
MPSO	Modified particle swarm optimization
N_{ng}	Total Number of Negative Test Sequences
N_{po}	Total Number of Positive Test Sequences
N_{un}	Total Number of Negative Test Sequences
N_{up}	Total Number of Positive Test Sequences
NF	Neuro Fuzzy
NN	Neural Network
NSGA	Non-dominated Sorting Genetic Algorithm
OLS	Orthogonal Least Square
PAM	Point Accepted Mutation
PCA	Principal Component Analysis
PCNSA	Principal Component Null Space Analysis
PDF	Probability Density Function
PFAM	Database of Protein Families
PIR	Protein Information Resource Database
PIR-ALN	Database of Curated and Annotated Protein Sequence Alignments
PNN	Probabilistic Neural Network
PRODOM	Protein Domain Families
PROSITE	Protein Database by Swiss Institute of Bioinformatics
PROTOMAP	Protein Topography and Migration Analysis Platform
PROTONET	Protein Network
PSI-BLAST	Position-Specific Iterated BLAST
PSO	Particle Swarm Optimization

RBFN	Radial Basis Function Network
RNA	Ribonucleic Acid
ROC	Receiver Operating Characteristic
ROLSA	Recursive Orthogonal Least Square Algorithm
SCOP	Structural Classification of Proteins Database
SGERD	Steady State Genetic algorithm for Extracting Fuzzy Classification Rules from Data
SLFN	Single Hidden Layered Feedforward Network
SOM	Self Organizing Map
SVD	Singular Value Decomposition
SVM	Support Vector Machine
T	Thymine
TN	Number of True Negative
TP	Number of True Positive
U	Uracil
UNIPROT	Universal Protein Resource
V	Loss Function
WTA	Winner Takes All
WTM	Winner Takes Most

List of Figures

List of Tables

List of Symbols

λ	Eigen Values
ϕ	Kernel Function to Map Input Space to Feature Space
σ	Gaussian Spread of RBF Kernel
P_c	Probability of Crossover
P_m	Probability of Mutation
α	Momentum in Backpropagation Algorithm
η	Learning Rate in Backpropagation Algorithm
ρ	Vigilance Parameter in Fuzzy ARTMAP
β	Learning Rate in Fuzzy ARTMAP
$\oplus$	Fuzzt T-Norm Operator
$\forall$	For all
ϵ	Belongs to
$\cup$	Union
∞	Infinity
Σ	Summation
ξ_i	Margin Error of Misclassified Points in SVM
γ	Variance of Gaussian RBF Kernel
α_i	Minimum Range of the Parameter
β_i	Maximum range of the Parameter
γ_i	Number of Precision Required after Decimal Point
n_i	Number of Bit in the Chromosome String
$\bar{T}_i$	Complement of T_i
$\|\ \|$	Norm Operation

Chapter 1

Introduction

1.1 Introduction

Bioinformatics is one of the leading research areas which integrates various fields such as advanced computer science and informatics, biology, statistics, applied mathematics, artificial intelligence, etc. to solve the biological problems at the molecular level. Application of advanced statistical and data mining techniques in the area of bioinformatics help to organize, analyse and interpret biological data and thereby discover previously unknown patterns. The major areas of Bioinformatics concern primary genome sequence, protein structure, micro-array and gene regulatory networks. The genome provides only static information whereas the gene expression patterns produced from the micro-array experiments provide dynamic information about cell function [1].

Analysis and interpretation of biological sequence data are fundamental task in bioinformatics. Classification and prediction techniques are one way to deal with such a task [2]. The problem of protein superfamily classification is a major research area of bioinformatics. Proteins are the building blocks of all living organisms. These are macro molecules which consists of carbon,

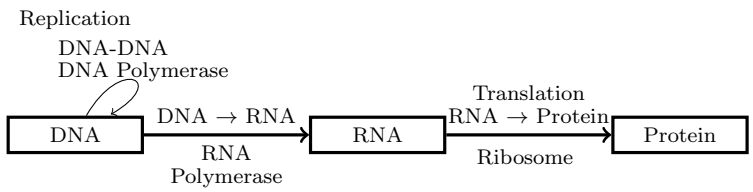

Figure 1.1: Schematic representation of Central Dogma of Life

hydrogen, oxygen, nitrogen, and sulphur atoms. The 20 different amino acids linked through peptide bonds are arranged in various combinations to generate a huge number of proteins. Some important functions of proteins include enzymes, hormones, antibodies etc. The Central Dogma of Life clearly describes the *formation of protein* inside a living organism. This is shown in Figure 1.1.

The Central Dogma consists of three main phases:

1. Replication: Deoxyribonucleic acid (DNA) gets duplicated by a process called replication prior to the occurrence of cell division. The replication of DNA allows each daughter cell to contain a full complement of chromosomes.

2. Transcription: This is the process of conversion of DNA of chromosome to form Ribonucleic acid (RNA). RNA contains ribose sugar where as DNA contains deoxyribose sugar. In addition, RNA lacks the base T. It is replaced, instead, with the base U, which is complementary to A (as T is complementary to A in DNA) (A = adenine, C = cytosine, T = thymine, G = guanine and U = uracil). The RNA formed acts as a messenger, which passes from the nucleus into the cytoplasm of the cell. So, this type of RNA is often called messenger RNA or mRNA.

3. Translation: The information now in the RNA sequence is decoded to form protein. This process is called translation.

The application of computational intelligence techniques to the field of bioinformatics can handle huge amounts of biological data and retrieve valu-

able knowledge from it. The majority of problems in bioinformatics are computationally hard in nature, and soft computing techniques offer promising approach to find solutions to these type of problems. The general problems in area of bioinformatics involve gene finding and promoter identification in DNA sequences, Gene regulatory network identification, DNA and RNA structure prediction, Protein structure and function prediction by superfamily classification, Gene mapping on chromosomes, etc.

Computational intelligence is a combination of three main paradigms such as Neural Networks, Genetic Algorithm and Fuzzy Logic. Evolutionary computation, swarm intelligence, probabilistic reasoning, etc. are few techniques integrated with computational intelligence techniques. Researchers have investigated that integration of various techniques such as neuro-fuzzy systems, evolutionary-fuzzy systems, evolutionary neural networks, evolutionary neuro-fuzzy systems, etc. have shown promising results in many real-life applications. This is due to their ability of tolerance for imprecision, uncertainty, approximate reasoning and partial truth. These intelligent techniques possess the real challenge to handle and manipulate the biological data as they are quite adaptive to changing environment. Besides that, the biological data have many missing and noisy samples, and the intelligent techniques are highly robust to handle these sort of data. Intensive work in the direction of protein secondary prediction using NN, protein functional prediction using SVM, protein tertiary structure prediction using NN, GA and SVM; protein docking using GA, etc. have already been implemented by researchers [1]. Their results are quite promising, which prove the phenomenal performance of intelligent techniques in the area of application to bioinformatics.

Identifying the structure and function of new proteins is the primary objective of the researchers working in the area of proteomics. Proteomics is the study of proteomes that includes determining 3D shapes of proteins, their role inside cells, the molecules with which they interact and defining which category of proteins are present and how much of each are present at given time. A

proteome is the complete collection of proteins within a cell or tissue or organism at a particular time [3]. Proteins are long strings of amino acid sequences, and the occurrence and combination of amino acids contributes for the correct prediction of structure and function of a newly discovered protein. Proteins are grouped into different families with significant sequence similarity showing 30% or greater common evolutionary relationship. Proteins are grouped into superfamily having low sequence similarity but possessing structural and functional features suggesting a common evolutionary origin. As the total number of sequenced proteins increases, and interest expands in proteome analysis, there is an ongoing effort to organize proteins into families and predict their family membership. Correct prediction of unknown protein or newly discovered protein mainly concerns the researchers and practitioners for prediction of molecular function, drug discovery, medical diagnosis, genetic engineering, etc. Protein classification can be done by classifying a new protein to a given family with previously known characteristics. The aim of classification is to predict target classes for given input protein. There are many approaches available for classification tasks, such as statistical techniques, decision trees and neural networks.

1.2 Data Mining in Proteomics using Intelligent Techniques

Data mining is defined as, exploration and analysis by automatic and semi-automatic means of large quantities of data, in order to discover meaningful patterns and rules. The data mining techniques combine the study from various areas such as statistics, database, machine learning, pattern recognition and optimization techniques. The application of data mining techniques has manifold tasks in the area of proteomics. The applications of various intelligent techniques offer promising solutions to the various problems in the area

of proteomics.

Classification of newly discovered protein to their superfamily for structure and function prediction is an important task in the area of proteomics. Fuzzy ART Map classifiers, ANN, SVM and Extreme Learning Machine (ELM) are implemented for protein superfamily classification. The protein structure prediction involves predicting the secondary structure state for each amino acid residue. The secondary structure of protein has three regular forms such as alpha helix, beta sheet and loop. The accuracy index used here is $P_{accuracy} = \frac{(P_\alpha + P_\beta + P_{loop})}{T} * 100$, where T is the total number of residues, P_α is the number of correctly predicted residues in α helix, P_β is the number of correctly predicted residues in β sheet and P_{loop} is the number of correctly predicted residues in loops [4]. ANN, Neuro-GA and SVM are successfully implemented for protein secondary structure prediction. The tertiary structure of protein is the stable 3-D structure that forms a polypeptide, and the function of a protein is determined from its 3-D shape or fold or conformation. The determination of an optimal 3-D conformation of a protein corresponds to folding, and has manifold implications to drug design [1]. ANN, GA and SVM are implemented for protein tertiary structure prediction and GA, Evolutionary Programming (EP) and SVM are used for protein fold detection.

Motif is a sequence of amino acids or nucleotides that performs a particular function and is often conserved in particular region. ANN, Neuro-Fuzzy, and Genetic Programming are implemented for motif identification and classification. Docking is frequently used to predict the binding orientation of small molecule drug candidates to their protein targets in order to predict the affinity and activity of the small molecule. Hence docking plays an important role in the rational design of drugs [5]. Drugs are ligands or enzymes that bind to an active site of protein. Docking can be categorized as: 1) rigid docking: both ligand and protein are rigid; 2) flexible-ligand docking: ligand flexible and protein rigid; and 3) flexible-protein docking: both ligand and protein are flexible. GA is applied for prediction of active sites in docking [1].

Phylogenetic analysis is performed to trace the evolutionary relationship of genes, proteins or species. NN and GA are used to predict the common evolutionary relationship. Protein homology detection is used to classify proteins into functional or structural classes by homologies. Detecting homologies at low levels of sequence similarity is remote homology detection. SVM is implemented for homology detection [6, 7].

1.3 Protein Superfamily Classification and its Importance

The problem of protein superfamily classification can be stated as, given a newly discovered amino acid sequence, responsible for the cause of a disease, the main task of the biologist is to classify the sequence to an existing superfamily. This helps in predicting the protein function and/or structure of the unknown sequence; thus avoiding the expensive biological (wet) experiments at 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 for treatment of diseases by discovering new drugs [8].

The major application of protein superfamily classification is in the area of drug discovery. If a newly discovered protein, responsible for the cause of a disease gets correctly classified to its superfamily, the task of the drug analyst becomes simpler. The analyst can perform molecular docking, which can be thought of as a problem of lock-and-key, where one is interested in finding the correct relative orientation of the key which will open up the lock. Here, the protein can be thought of as the lock and the ligand as a key [9]. The ligand database can be searched for all possible orientations and conformations of the protein belonging to that superfamily paired with the ligand. Thus, the search space is reduced enormously as the protein in the given protein-ligand pair is

searched under a particular protein superfamily.

1.4 Protein Superfamily Classification as a Problem of Pattern Classification

The problem of protein superfamily classification can be mapped as a pattern classification problem. The long strings of amino acid sequence represent a pattern from which many global and local features are extracted. The features selected or extracted using filter and wrapper approaches help in classification of protein to their superfamily for structure and function prediction. Pattern classification refers to the task of placing some object to correct class based upon the measurement about the object [10]. The main task in building a pattern recognition system is to automate a machine using some machine learning techniques so that it can receive patterns as input and correctly classify them into respective classes. Tom Mitchell defined machine learning as, a computer program that is said to learn from experiment E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E [11]. The four best-known approaches for pattern recognition are: 1) Template matching, 2) Statistical classification, 3) Syntactic structural matching and 4) Neural networks. In template matching, the pattern to be classified is matched against the stored template, whereas in statistical classification, each pattern is represented in terms of 'd' features and a discriminant analysis based approach is used for classification. In syntactic approach, a formal analogy is drawn between the structure of patterns and the syntax of a language and in neural networks based classification, the network learns the complex input-output relationships and converges to meet a certain threshold mean square error value [12]. The major steps of a pattern classification system are shown in Figure 1.2.

1.4.1 Basic Steps of Pattern Classification System

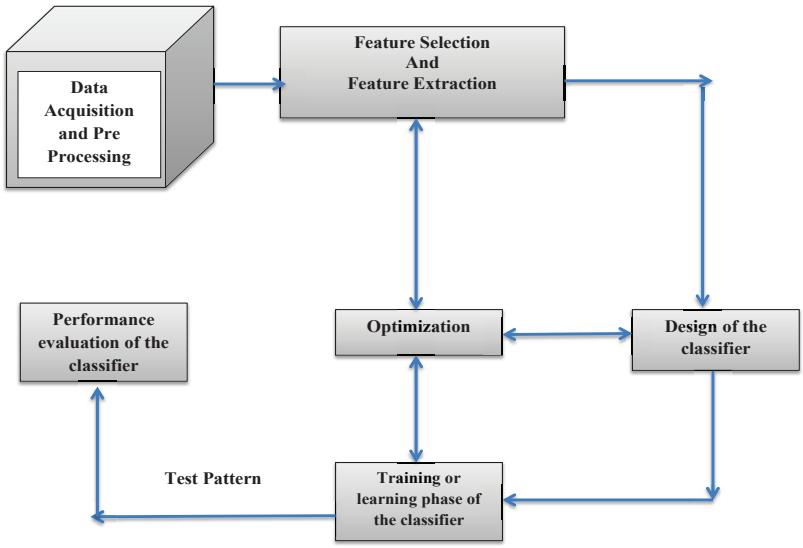

Figure 1.2: Basic steps of a pattern classification system

- Data acquisition and preprocessing

  The first step of a pattern classification system is data acquisition in which the raw data are derived or collected from the source such as from sensors, cameras, databases, etc. The data so derived may be incomplete, noisy (containing errors and outlier values that deviate from the expected) and inconsistent (containing discrepancies). In data preprocessing, the data having missing values are filled, smoothing of noisy data are done, outliers are removed and inconsistencies are resolved.

- Feature selection and feature extraction

  The main objective of feature selection or extraction is to select a subset of 'm' features out of 'd' number of features while maintaining an optimal level of classification accuracy. In feature selection, a subset of features are selected based on some measures where as in feature extraction, 'd'

dimension feature vector is reduced to 'n' dimension using some linear or non-linear transformation techniques. The linear transformation techniques mostly used are PCA, Singular Value Decomposition (SVD), Independent Component Analysis (ICA), Linear Discriminant Analysis (LDA), etc. Feature extraction techniques for non-linearly distributed data are Kernel PCA (KPCA), Multidimensional scaling (MDS), Gaussian process latent variable models, etc.

- Learning of the classifier (Machine learning)

  Machine learning techniques can be of three types in which the machine learns and gets adapted to the training patterns. In *supervised learning*, the data in the training patterns are associated with a class label or target vector, and learning continues with an objective to reduce sum of mean square error (MSE) costs over the training patterns. In *unsupervised learning or clustering*, learning takes place with unlabelled data. The system learns of its own forming clusters or natural grouping based on the commonality of features in the data. The third category is *reinforcement learning* or *learning with a critic*, in which no desired category symbol is given, instead; the only teaching feedback is that the tentative category is right or wrong.

- Optimization

  Optimization is combined in almost all the stages of a pattern recognition system. In preprocessing, optimization guarantees that the best quality input features are derived by removing noise from the background source of the input. In feature selection and feature extraction stage, selection of an optimal number of distinguishing features greatly affects the performance of the classifier. In the classifier design phase, structure optimization of the network is considered and in the learning phase, optimization of synaptic weights and other parameters are done with an overall objective to decrease the mean error rate value.

- Performance evaluation of the classifier

  To measure the performance of the classifier; classification accuracy is evaluated, which gives the total number of samples correctly classified with respect to the total number of test samples. Based on specific applications, some other parameters, such as Sensitivity, Specificity, Receiver operating characteristic (ROC) Curve, False acceptance rate (FAR), False reject rate (FRR), etc. are evaluated.

1.5 Motivation

Proteins are the cause of many diseases. If a newly discovered protein gets correctly classified to its superfamily, then the task becomes easy for the drug analyst to discover new drugs. The analyst may re-combine some existing drugs or start searching the ligand database paired with that protein superfamily. In this way, the analyst may be successful in finding out the right ligand for the new protein. Therefore, correct classification of proteins becomes a very challenging task as it guides the analysts to discover appropriate drugs. But as the protein sequence is of high dimension, and also contain missing and noisy samples, soft computing techniques can be correctly applied for the problem of protein superfamily classification. The soft computing techniques are robust, and possess the ability of tolerance for noise, imprecision, uncertainty and approximate reasoning.

1.6 Objective

The main objective of the present work is to develop an efficient classifier showing high level of performance accuracy. Since the selection of significant features has a great impact on the performance of the classifier, efforts are made to extract optimal number of distinguishing features. The main focus given in this thesis work are on two main aspects namely feature extraction

and classifier design. These two aspects are implemented using different computational intelligence techniques such as neural network, genetic algorithm and support vector machine.

1.7 Contributions

In order to proceed with the task of protein superfamily classification, a detailed investigation was done on various features extracted from amino acid sequence and the various classifiers implemented by earlier researchers. SVD and PCA were implemented for dimension reduction of long feature vector derived from every amino acid sequence. As the performance of PCA was better than SVD, so in subsequent chapters PCA and some modification to PCA were used for feature extraction. From the implementation of standard neural networks, it was observed that FFNN and PNN have some shortcomings. So, RBFN was preferred over FFNN and PNN.

Though RBFN was selected to perform classification task, the major focus was given for structure optimization and improvement of performance accuracy of RBFN. A variation to the concept of MOGA i.e., Adaptive MOGA (AMOGA) was proposed for improving the convergence rate of MOGA. The optimized structure of RBFN was derived from the pareto optimal set obtained after the implementation of AMOGA.

The next contribution of the thesis was on significant feature extraction using the proposed algorithm PCA-NSGA-II. This algorithm searches the eigen space and selects the eigen vectors which have a great impact on the performance of the classifier. To derive the most parsimonious architecture of RBFN having high performance accuracy, RBFN-ROLSA was implemented.

Neural networks suffer from major drawbacks like problem of local minima, high computational burden and over-fitting. So, SVM was preferred to perform classification. The subsequent contribution of the thesis is the implementation of MOGA-SVM, which optimizes the number of eigen vectors as well as the

hyper-parameters of SVM model. To improve the convergence rate of MOGA, AMOGA-SVM is implemented. Keeping the stopping criteria constant for both approach, a comparative study was performed.

1.8 Organization of Thesis

The rest of the thesis is organized as follows:

Chapter 2 provides insight on the state-of-art of various techniques applied for protein superfamily classification problem. The review has been done in two broad parts with respect to the objectives. First part describes the various methods of feature extraction and feature selection from amino acid sequences. Second part describes the design of classifier for classifying proteins to their superfamily. The subsequent section describes the various performance measures used for the task of protein superfamily classification. Some existing techniques are implemented and the results of few numerical simulations are shown. Dimension reduction techniques, classification using standard neural networks and few evolutionary optimization techniques applied for optimizing the structure of feed-forward neural networks are implemented in this chapter.

Chapter 3 describes the steps for implementation of PCA for dimension reduction and significant features extraction from long feature vector of amino acid sequences. AMOGA a variation of MOGA is applied for the structure optimization of RBFN. The detailed steps of AMOGA is described in this chapter. A comparison between both the approaches i.e., MOGA and AMOGA are done by performing numerical simulations. The performance of RBFN-MOGA and RBFN-AMOGA are compared with standard neural networks.

Chapter 4 describes the two stage approach for protein superfamily classification. In the first stage, optimal number of features are extracted using PCA-NSGA-II (non-dominated sorting GA) and in the second stage, Recursive Orthogonal Least Square Algorithm (ROLSA) in used to train RBFN. ROLSA is used for structure optimization of RBFN as well as derives the optimal value

of the weight matrix connecting the hidden and the output layer. The detailed steps of PCA-NSGA-II and RBFN-ROLSA are described in this chapter. The experimental details and the results obtained from numerical simulations are discussed here.

Chapter 5 describes the MOGA approach to select the optimal number of significant eigen vectors from the eigen space as well as optimize the hyper-parameters of SVM. MOGA-SVM, selects the non-dominated solutions obtained from the pareto front to solve the trade-off problem between the number of eigen vectors selected and the accuracy obtained by the SVM classifier. The steps of MOGA-SVM are shown in flow chart and the detailed steps are outlined in the algorithm. To improve the convergence rate of MOGA-SVM, AMOGA-SVM is implemented and a comparative study between MOGA-SVM and AMOGA-SVM is performed.

Chapter 6 concludes the thesis. In this chapter, the work done is summarised, the contributions are highlighted and suggestion for the future work has been discussed.

Chapter 2

Related Work

This chapter focusses on the state-of-art of various techniques applied by the researchers for protein superfamily classification problem. The review has been done in two broad parts with respect to the objectives of the thesis. First part describes the various methods of feature extraction and selection from amino acid sequences. Second part describes the design of classifier. In the subsequent section, the various parameters used for measuring the performance of the classifier are listed.

2.1 Introduction

Although, many trivial alignment methods are already developed by earlier researchers, but the present trend demands the application of computational intelligent techniques to perform the task of protein superfamily classification. Earlier approaches used *sequence similarity* concept for protein superfamily classification. These includes Smith Waterman [1981], FASTA [Pearson, 1990], BLAST [Altschul et al., 1997], PSI-BLAST [Altschul et al., 1997]. In these approaches, two protein sequences are taken as input and the similarity measure is calculated between them. BLOSUM [Henikoff, 1992], PAM [Dayhoff et al.,

1978], are most commonly used scoring matrices which are used to derive alphabet weighted similarity. Classification based on *motifs and domains* assume, domains form the building blocks of proteins. Motifs are composed of sub-strings occurring in local regions of a sequence. PROSITE [Falquet et al., 2002] is the oldest motif-based method of classification of proteins. Other than PROSITE, some more classification systems developed are BLOCKS [Henikiff et al., 2000], PFAM [Bateman et al., 2000] based on Hidden Markov Model (HMM), PRODOM [Corpet et al.,2000], EMOTIF [Atwood et al., 2002], etc. The two major drawbacks of domain based classification are; many proteins may have several domain appearances and there may be some protein which may don't have any domain. The software systems already developed based on full protein sequence includes PROTOMAP [Yona et al., 2000], PROTONET [Sasson et al., 2003], PIRALN [Srinivasrao et al., 1999]. Classification system based on *phylogeny* was developed in 2001 COGS [Tatusov et al.] performed clustering of proteins. The structure rather than sequence has a greater influence in predicting the functional properties of proteins. SCOP [LoConte et al., 2002] is structural classification of proteins which classifies proteins into four levels of hierarchy such as Family, Superfamily, Fold and Class. CATH [Orengo et al., 1999], FSSP [Holm and Sander, 1998], etc. are few classification systems based on protein structure. The main aim of protein superfamily classification is functional annotation and functional prediction of newly discovered protein sequence.

2.2 Feature Selection from Amino Acid Sequence

Choosing an appropriate set of relevant features is a critical issue for any pattern classification problem. The main objective of feature selection is to select *'m'* number of distinguishing features out of total *'n'* number of features

such that $m \ll n$. For any feature subset selection method, the most important factors need to be considered are evaluation measure and the search strategy [13]. The primary goal of feature selection are *reduction of cost of extracting features* and *improving the performance of the classifier*. Typical evaluation measures for feature subset selection can be divided into *filter and wrapper* based approaches. Feature subset evaluation using a learning algorithm by implementing a classifier is wrapper based approach whereas evaluating the goodness of selected features using certain criteria is filter based approach [14].

A survey on evaluation functions used in inductive algorithm is shown in Ben-Basset's survey in [15].

2.2.1 Global Feature Selection from Amino Acid Sequence:

The frequency occurrence of 2-gram or bi-grams of any two amino acids occurring consecutively and also the consecutive occurrence of any two exchange groups are derived as global features from amino acid sequence. The exchange groups statistically describes the probability of one amino acid replacing another over time representing high evolutionary similarity [16]. The second most vital global feature is the correlation coefficient which measures the global correlation structure of the given sequence compared to the sequence belonging to the target family.

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 [17].

Bi-gram feature value

The i-th bi-gram feature value v_i is calculated as :

$$v_i = \frac{f_i}{\mid S \mid -1} \tag{2.1}$$

where $1 \leq i \leq 436$ (400 represents 2-gram features derived from twenty amino acid bases and 36 denotes the 2-gram feature derived from six exchange groups). Here the denominator denotes the number of bi-grams possible in a sequence of length $\mid S \mid$ and v_i denotes the proportional frequency of occurrence of i-th bigram feature (f_i).

Mean and standard deviation

$$Mean(\bar{v_i}) = \frac{\sum_{j=1}^{N} v_{ij}}{N} \tag{2.2}$$

The standard deviation can be calculated as :

$$Std.\ Dev.(s_i) = \sqrt{\frac{\sum_{j=1}^{N}(v_{ij} - \bar{v_i})^2}{N-1}} \tag{2.3}$$

If $\bar{v_i}$ denotes the mean feature value of i-th bi-gram feature, then for 436 features, 436 mean feature values are obtained such as $\bar{v_1}, \bar{v_2} \cdots \bar{v_{436}}$ and v_{ij} denotes feature value at index (i,j). This concept is implemented in [16, 18].

Correlation coefficient measure

The correlation coefficient measure denoted as $CC(S_j)$ compensates for the loss of information for not considering all the bi-gram features as inputs to the classifier [16, 18]. It is calculated as:

$$CC(S_j) = \frac{436 \sum_{i=1}^{436} v_{ij}\bar{v_i} - 436 \sum_{i=1}^{436} v_{ij} \sum_{i=1}^{436} \bar{v_i}}{\sqrt{(436 \sum_{i=1}^{436} v_{ij}^2 - (\sum_{i=1}^{436} v_{ij})^2)(436 \sum_{i=1}^{436} \bar{v_i}^2 - (\sum_{i=1}^{436} \bar{v_i}^2))}} \tag{2.4}$$

Position specific encoding

Wu et al. had proved that the encoding method has a great impact on the performance of the classifier in [19,20]. For every n-gram feature, the frequency of occurrence (count) and position can be obtained. The order of occurrence is mostly not taken into consideration. Thus, each n-gram pattern can be represented in either of three ways: such as, count vector only, position vector only or concatenation of both vectors.

2.2.2 Feature Selection using Hydropathy Content

Shakir Mohamed et al. considered the hydropathy properties of every amino acid sequence and derived 18 features as a measure of this property [21]. The hydropathy property describes amino acids to be either of three types such as hydrophobic, hydrophilic (polar) or neutral. The Chothia and Finkelstein [22] calculates three descriptors for hydropathy classification such as hydropathy composition(C), the hydropathy transmission(T) and the hydropathy distribution(D). The composition value(C) gives three values which is calculated as the frequency of hydrophobic, hydrophilic (polar) and neutral amino acids in the sequence. The transmission value(T) gives three values where the number of times a polar molecule is followed by a neutral molecule or vice-versa, similarly for hydrophobic followed by hydrophilic and vice-versa and neutral followed by hydrophobic and vice-versa. The hydropathy distribution(D) is calculated as the frequency of hydrophobic, hydrophilic and neutral molecules at each interval of 25%, 50%, 75% and 100% of the amino acid sequence. This results in 12 features, 4 features for each of the three hydropathy groups. Besides these 12 features, other six features (3 for C and 3 for T values) are also calculated, thereby resulting a total of 18 features.

2.2.3 Feature Subset Selection using Relative Entropy

Feature selection using relative entropy measure for eight protein superfamilies is described in [23].

If X_j is the feature and c_{ij} be the occurrence number of the feature X_j in the sequence S_i then the frequency f_j for feature X_j can be defined as:

$$f_j = \frac{\sum_{i=1}^{N} c_{ij}}{\sum_{i=1}^{N} \sum_{j=1}^{436} c_{ij}} \tag{2.5}$$

where N is the total number of sequences in the target or the non-target class.

Let $P(x \mid t)$ denotes the class conditional density functions for feature X, over

the target class

$P(x \mid nt)$ denotes the class conditional density functions for feature X, over the non-target class.

Let W(x) is the relative entropy function between $P(x \mid t)$ and $P(x \mid nt)$. W(x) may be evaluated as:

$$\begin{aligned} W(x) = \quad & -\sum P(x \mid t) logP(x \mid t) - \sum P(x \mid nt) logP(x \mid nt) \\ & + \sum P(x \mid t) logP(x \mid nt) + \sum P(x \mid nt) logP(x \mid t) \end{aligned}$$

Smaller W(x) values indicates greater distinction between the two classes.

2.2.4 Feature Subset Selection using Distance Measures and Feature Ranking

In order to select subset of features out of long feature vector, the distance measure is used to derive features having maximal discrimination power [24]. It can be calculated as :

$$D(v_i) = \frac{(\bar{v_{i+}} - \bar{v_{i-}})^2}{s_{i+}^2 + s_{i-}^2} \tag{2.6}$$

where $(.)_+$ and $(.)_-$ refer to values of the measure calculated over the positive and negative training data sets respectively. $\bar{v}_i$ and s_i are the mean and standard deviations of the i-th feature in the feature vector. The bi-gram features having highest D(v_i) values are selected. The D(v_i) values are sorted in descending order and the best features are selected based on their rank. This technique is otherwise known as Feature Ranking Algorithm which is implemented by Mansoori et al. $D(v_i) = max(D_1, \cdots, D_j \cdots, D_{M-1})$. The first E' out of E features are selected from the ranked list denoted as $(f_1 \cdots f_{E'})$ where $f_1 \geq \cdots \geq f_i \geq \cdots \geq f_E$ and f_i is a unique feature label in 1, . . . , E.

2.2.5 Feature Subset Selection using Statistical Profiles

S. Bandyopadhyay proposed a new technique for protein feature extraction using statistical profile which is based on the concept of inheriting features by primary structure of proteins from their ancestors [25]. A statistical profile is constructed using a 20×lmax probability matrix where, lmax is the maximum length of a sequence belonging to a particular superfamily. The position (i,j) indicates the probability of occurrence of the i-th amino acid in position j of the sequence. From the statistical profile, position specific weight of any amino acid in a given sequence can be obtained, by adding the occurrence 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.

2.2.6 Feature Subset Selection using Genetic Algorithm

Genetic Algorithm (GA) is a randomized evolutionary heuristic search technique which have been successfully applied for selecting optimal number of significant features. A novel approach for optimizing features and training of RBFN is implemented using GA [26]. This approach has outperformed earlier approach of BLAST and the HMMer for protein sequence classification obtained from Protein Information Resource (PIR) database.

The two critical issues of GA are: encoding of chromosome and designing of the fitness function. The chromosome is an initial probable guess for a solution to the problem which should be encoded correctly. For feature selection, it is generally encoded as strings of 0's and 1's where 1 represents inclusion of the feature and 0 indicates the discard of the feature as shown in Figure 2.1.

The two main objectives of this classification problem are, to maximize the classification accuracy of the classifier and minimize the number of features. The trade-off between the two objectives can be mapped into a single objective function representing a weighted sum of objectives such as:
$f(x) = w1 * f_1(x) + w2 * f_2(x)$, where, w1 and w2 are weight coefficients and

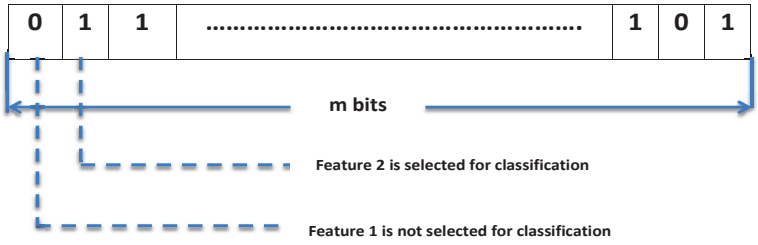

Figure 2.1: Representation of chromosome for feature selection

f_1(x) is the recognition rate and f_2(x) is the number of features removed from the feature set.

Zhao et al. have demonstrated the application of GA for selection of the eigen vectors from the covariance matrix in [27]. GA selects the best reduced global and local features from composition and motif content and optimizes the regularization parameter of SVM simultaneously. After feature extraction, SVM is used as a classifier and this approach has proven effective for protein superfamily classification. A detailed description of dimensionality reduction using GA is shown in Figure 2.2. The numerical simulation using the above technique on various data sets is shown in [28].

2.2.7 Local Features Selection using Motif Content

Motifs are local features or the conserved region in the amino acid sequence which signifies structural and functional biological properties. Based on the local interactions of amino acids and exchange groups, the local features can be extracted from the sequence. Blekas et al. applied a unsupervised motif discovery algorithm for class dependent and class independent motifs to identify the probabilistic motifs in [8]. The discovered motifs are then converted to a real valued input vector which is given as input to the feedforward neural network. Wang et al. derived the local similarity measure (LS) from the motif content of the amino acid sequence in [18].

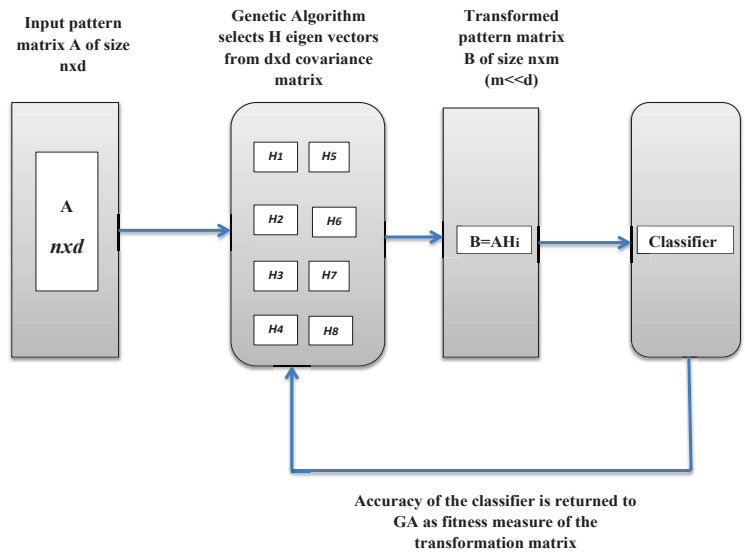

Figure 2.2: Feature extraction using Genetic Algorithm

2.2.8 Features Directly Extracted from Amino Acid (Based on the Properties)

The features directly extracted from amino acid based on the properties is shown in [29].

- **Atomic composition:** Counts the Carbon, Hydrogen, Nitrogen, Oxygen and Sulphur atoms in the amino acid sequence.
- **Molecular weight:** Mass of a molecule of a substance IS based on 12, as the atomic weight of carbon is 12. It is calculated in practice by summing the atomic weights of the atoms making up the substance's molecular formula.
- **Isoelectric point:** The isoelectric point (pI) is the pH at which a particular molecule or surface carries no net electrical charge. The net charge on the molecule is affected by pH of their surrounding environment and can become more positively or negatively charged due to the loss or gain of protons (H^+). At a pH below their pI, proteins carry a net positive

charge and above their pI they carry a net negative charge. Proteins can thus be separated according to their isoelectric point (overall charge).

- **Length of amino acid sequence:** There are twenty standard amino acid bases for a protein sequence. The sum of individual frequency of occurrence of every amino acid base gives the length of the protein sequence.

- **Average Mass of Protein Sequence:** The average mass of a molecule is obtained by summing the average atomic masses of the constituent elements. For example, the average mass of natural water with formula H_2O is 1.00794 + 1.00794 + 15.9994 = 18.01528.

- **Nominal Mass of Protein Sequence:** The nominal mass of an ion or molecule is calculated using the integer mass (ignoring the mass defect) of the most abundant isotope of each element. This is equivalent to summing the mass numbers of all constituent atoms. For example H = 1, C = 12, O = 16, etc. The nominal mass of water is 18, for example.

2.3 Feature Extraction using Dimension Reduction

Feature extraction algorithms are the methods or techniques that create new features based on transformations or combinations of the original feature set. In other words, given a $n \times d$ pattern matrix A (n points in a d-dimensional space), a $n \times m$ pattern matrix B is being derived, such that $m << d$ where $B = AH$ and H is a $d \times m$ transformation matrix [28].

2.3.1 Feature Extraction using Singular Value Decomposition (SVD)

Singular value decomposition (SVD) technique was used to reduce the dimension of large sparse n-gram feature matrix, implemented by Cathy Wu et al. in [17]. SVD reduced the size of feature vector showing an overall performance of 90% sensitivity value. PCA, SVD etc. are most commonly used techniques for multivariate data such as Gene, Microarray, Protein, etc.

2.3.2 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 feature space of high dimension into a feature space of lower dimension having the most significant features. The implementation of PCA for feature extraction is implemented in [30].

To investigate the performance of SVD and PCA, a comparative study of SVD and PCA was performed on protein data using PNN as classifier. The three superfamilies considered in experiment are Esterase (145), Lipase(155), Cytochrome(140) from UNIPROT database (`http://www.uniprot.org/`). From each family, 70% of total data set formed the training set and the remaining 30% formed the test set. The comparison results obtained from numerical simulations are shown in graph (Figure 2.3). It was observed that, PCA performed better in comparison to SVD.

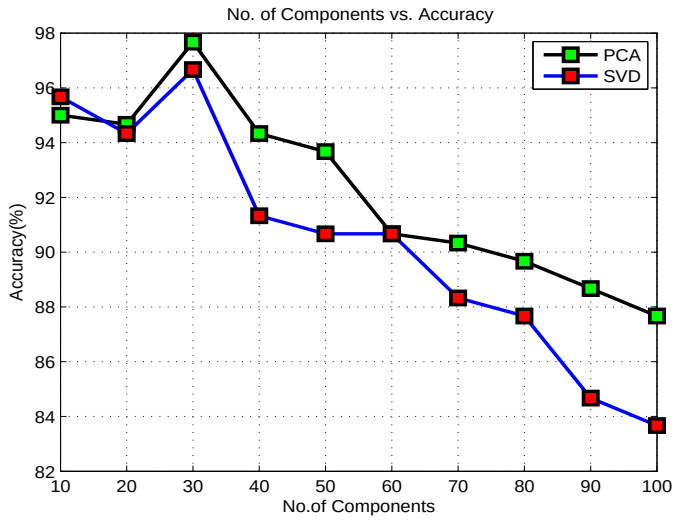

Figure 2.3: Performance of PCA and SVD using PNN as classifier.

2.4 Design of Classifiers

2.4.1 Design of Classifier using Neural Networks

Feedforward Neural Networks

In a typical multi-layered feedforward neural network (FFNN), neurons are organized into three layers (shown in Figure 2.4). The input layer is composed of neurons, which consists of the values in a data record, and that constitutes inputs to the next layer of neurons. The next layer is called hidden layer and there may be more than one hidden layer. The final layer is the output layer, where every node represents a class. A single sweep forward through the network results in the assignment of a value to each output node, and the given test input is assigned to that class node whose neural network output is very close to the target value. Multilayer feedforward networks are generally trained using the Backpropagation (BP) learning algorithm.

Backpropagation (BP) algorithm FFNN using BP training algorithm for protein superfamily classification is implemented in [17, 30]. In BP al-

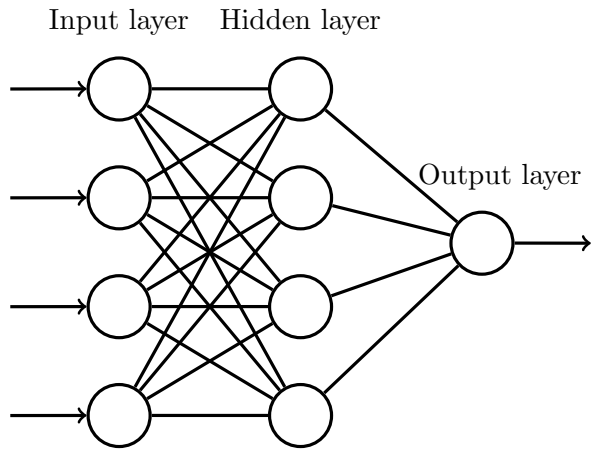

Figure 2.4: A typical multi-layered feedforward neural network

gorithm, functional signals flows in forward direction and error signals propagate in backward direction. That's why it is called Error Backpropagation or shortly backpropagation network. The activation function that can be differentiated (such as sigmoidal activation function) is chosen for hidden and output layer computational neurons. The major drawback of BP algorithm is, it takes long and uncertain training time and may get trapped in local minima. The rule for changing values of synaptic weights follows generalized delta rule but they are limited to searching for a suitable set of weights in an a priori fixed network topology. This mandates the selection of an appropriate optimized synaptic weight for the learning problem on hand. Many evolutionary optimization techniques can be applied to fasten the training process of the FFNN by deriving the optimal values of synaptic weights.

To investigate the performance of evolutionary algorithms, numerical simulations were performed on protein data set (The experiment details are same as described in Section 2.2.2).

- Genetic Algorithm (GA): GA is a stochastic based global searching technique which may be used to find out the optimized synaptic weight. Thus, a hybrid method combining GA-BP is implemented

and the predictive accuracy is calculated.

- Adaptive Genetic Algorithm (AGA)-BP: To overcome the limitations of GA such as premature convergence due to local optima and low convergence speed, an attempt has been made towards the improvement of parameters such as crossover probability and mutation probability. The probabilities of crossover and mutation are adaptively varied to protect the high fitness solutions from disruption as described in [31]. After implementation, it was observed that AGA-BP gave better result in comparison to GA-BP and traditional BP in terms of speed, predictive accuracy, and precision of convergence.
- Particle Swarm Optimization (PSO)-BP: PSO-BP encodes the parameters of neural networks as particles and the population of particles are referred as *Swarm*. The bias neuron is not included in the encoding of the particles. Here, the synaptic weights of the neural network are initialized as particles and the PSO is applied to obtain the optimized set of synaptic weights.
- Modified Particle Swarm Optimization (MPSO)-BP: In MPSO-BP, the probability of mutation is considered as 0.05 and the randomly generated particles undergo mutation. The training process is same as the PSO-BP, but a mutation phase is incorporated just before the completion of one generation.
- Differential Evolution (DE)-BP: It is a robust stochastic based search algorithm, for real parameter optimization. DE uses parameter vectors as individuals in a population. The key element distinguishing DE from other population based techniques is the use of differential mutation operator and trial parameter vectors.

From the graphs obtained from simulations (Figure 2.5(a) Mean Fitness vs. No. of Gens. and Figure 2.5(b) Accuracy vs. No. of Gens.), it is observed that AGA-BP has outperformed all other evolutionary

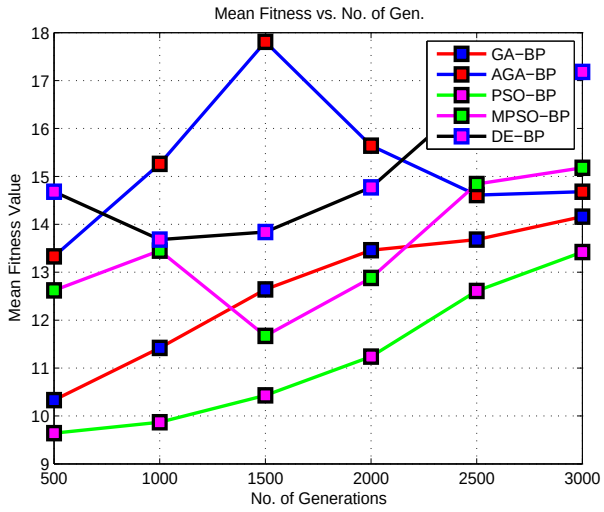

Figure 2.5: Mean Fitness vs. No. of Gens.

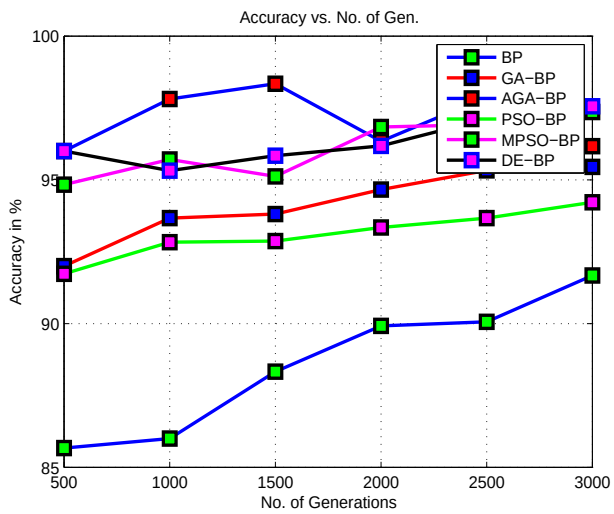

Figure 2.6: Accuracy vs. No. of Gens.

optimization techniques in terms of convergence rate and performance accuracy.

FFNN Trained using Kohonen's Unsupervised Learning Algorithm

In 1982, Teuvo Kohonen, developed *Self-organizing map*(SOM) which is a type of neural network. SOM are so named because the network undergoes unsupervised competitive learning to map the weights to the given

input data. This approach is based on *Winner takes all* (WTA) and *Winner takes most* (WTM) concept. When input pattern is presented, a distance to each neuron's synaptic weight is calculated. The neuron whose weights are most correlated to the current input vector is declared as the *winner.*

E.A Ferran et al. implemented ANN trained using Kohonen's unsupervised learning algorithm to cluster protein sequences into families in [30]. Bi-gram features extracted from 1758 protein sequences formed the input pattern matrix given to the network. Each protein pattern is presented as input to the network and the neuron having the closest synaptic vector to the input pattern is the winner neuron.

FFNN Trained using Gauss-Newton Bayesian Regularization (GNBR)

In the Bayesian regularization framework, the objective function is formulated as the weighted sum of two terms. They are:

1. the sum of squared error(E_x)
2. sum of squares of network weights

Using Bayes rule, the posterior probability distribution for the weights W of the network, given a training set X can be written as:

$$P(W \mid X) = \frac{P(X \mid W)P(W)}{P(X)} \tag{2.7}$$

By properly choosing the prior distribution P(W) and the likelihood function $P(X \mid W)$, the posterior distribution (Bishop, Foresec and Hagan) can be calculated.

The GNBR algorithm follows a Gauss-Newton approximation method implemented in [8] (Foresse and Hagan, 1997) for calculating the Hessian matrix at the minimum point using the Levenberg-Marquardt optimization algorithm.

Bayesian Neural Network (BNN)

The Bayesian Neural Network (BNN) basically has three layers namely, input layer, hidden layer and output layer. The number of nodes in input layer depends on the size of input feature vector. There may be multiple number of hidden nodes in the hidden layer and the output layer has a single output node. The output node is based on the logistic activation function such as $f(a) = 1/(1 + e^{-a})$. BNN is fully connected between the three layers. The architecture of BNN is shown in Figure 2.6.

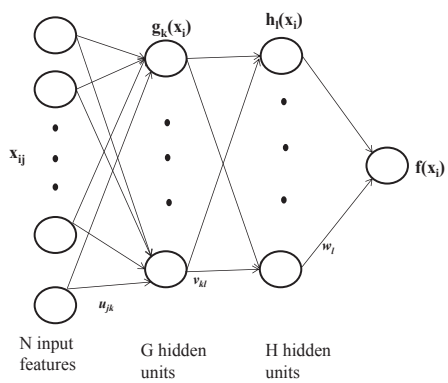

Figure 2.7: Architecture of Bayesian Neural Network

The Bayesian learning process is a three level inference process which undergoes iteration till the convergence criteria is met. The detail learning algorithm and application to protein superfamily classification are described in [18].

Radial Basis Function Network (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. The architecture of RBFN is shown in Figure 2.7.

In generalized RBFN, the supervised learning of the center 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. Protein

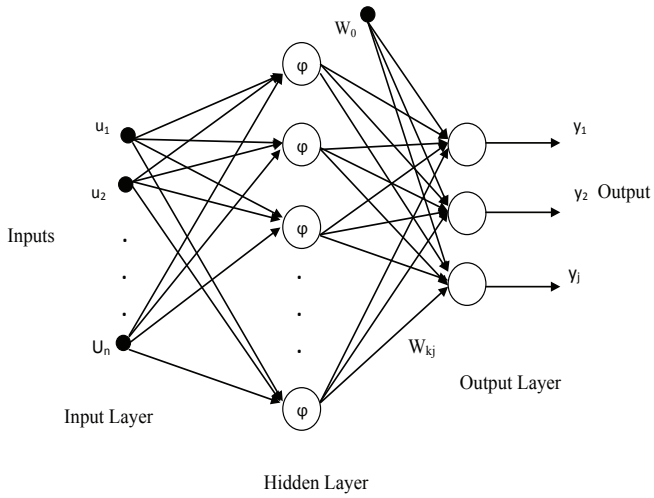

Figure 2.8: Architecture of Radial Basis Function Network

superfamily classification using modular RBFN with transition output fusion is shown in [32] and RBFN trained using Genetic Algorithm is implemented in [26]. An implementation of subtractive clustering in standard RBF and modular RBF is shown in [33]. The proposed network based on subtractive clustering has shown less training time compared to standard RBFN.

Probabilistic Neural Network

The concept of PNN was developed by Donald Specht in 1990 [34]. The concept of PNN relies on Parzen Window classifier. In original Specht's implementation, the basis function used as window is Gaussian Kernel which is given by:

$$g(x) = \frac{1}{n\sigma} \sum_{k=1}^{n} exp^{\frac{-(x-x_k)^2}{\sigma^2}} \tag{2.8}$$

where n= number of samples from a class

σ = smoothing parameter

x= unknown input

x_k is the "kth" sample.

The PNN is a multilayer feedforward network having four layers namely:

input layer, hidden or pattern layer, summation layer, output or decision layer shown in Figure 2.8. The pattern layer has one pattern node for each training sample. The summation node or unit receives the outputs from the pattern nodes associated with a given class. It simply sums the outputs from the pattern nodes that correspond to the category from which the training pattern was selected. Thus, the number of nodes in the summation layer is same as the number of classes in multi-class classification problem. The output node takes the decision of classifying the unknown sample to its respective class.

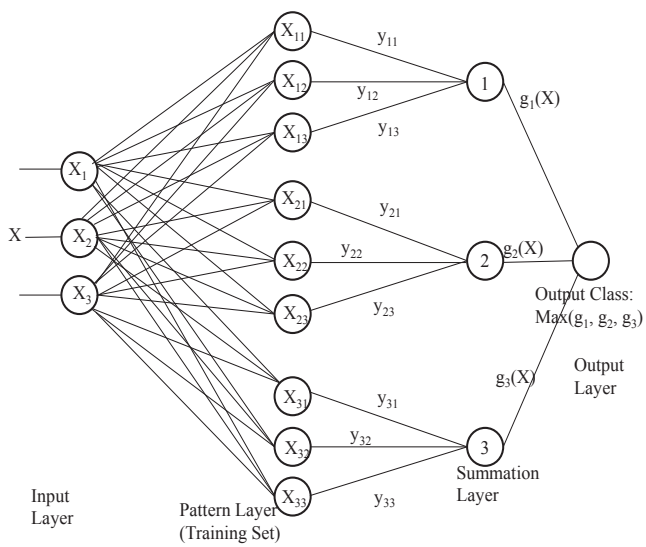

Figure 2.9: Architecture of Probabilistic Neural Network

The smoothing parameter value σ can be guessed, based on the knowledge of the data or the value which can be estimated using some heuristic technique. To classify the family membership of unknown proteins using PNN as a classifier is shown in [29].

To evaluate the performance of neural networks, a comparative study of three neural networks was done and the results obtained from numerical simulation is shown in graph (Figure 2.9). The experiment details are same as described in Section 2.2.2. It is observed that PNN has outperformed FFNN and RBFN.

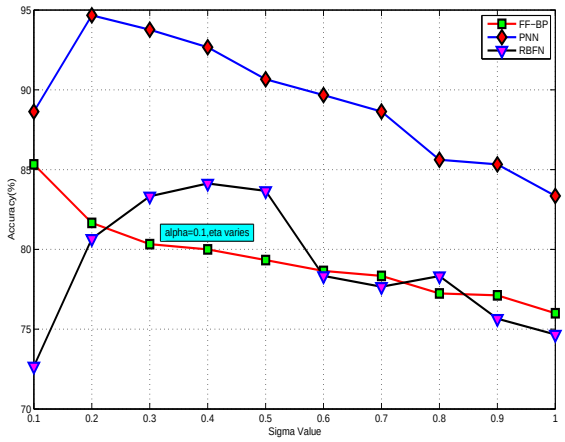

Figure 2.10: Performance of FFNN, RBFN and PNN

Extreme Learning Machine

In 2005, Huang et al. proposed a new learning algorithm called Extreme Learning Machine (ELM) for single hidden layered feedforward network (SLFN) with additive neurons and for kernel based radial basis function network. ELM does not have any control parameters i.e, stopping criteria, learning rate, learning epochs, etc. to be manually tuned and therefore it can be implemented easily. The application of ELM for protein sequence classification by Wang and Huang is implemented in [35].

2.4.2 Design of Classifier using Fuzzy Rules

The concept of Fuzzy Logic (FL) was developed by Lotfi A. Zadeh in 1965. FL is a multi- valued logic, that allows intermediate values to lie between 0 and 1. The concept of FL is widely used in many complex industrial processes, expert system, embedded system, electronics devices etc. The concept of FL is implemented to design fuzzy rule based classifier which is efficiently implemented for protein superfamily classification.

Designing Fuzzy Rule Based Classifier

Mansoori et al. designed a fuzzy rule based classification system which generates simple and comprehensible set of fuzzy classification rules based on distribution of amino acids from the training set [36].

A Steady State Genetic algorithm for Extracting Fuzzy Classification Rules from Data (SGERD)

SGERD, a novel steady state genetic algorithm is implemented for extracting compact set of simple and interpretable fuzzy classification rules from a dataset of protein superfamily sequences [24]. The main objective of SGERD is to generate a pre-specified number of Q rules per class (i.e., the best ones) in the final population for an n-dimensional problem with M classes and m labelled patterns.

Using ARTMAP

The implementation of Fuzzy ARTMAP is used as a classifier for protein superfamily classification in [21]. This classifier is based on adaptive resonance theory (ART). The learning system is built upon two fuzzy ART modules which employs calculus based fuzzy operations. The two controlling parameters are ρ and β which represents the vigilance parameter and the learning rate respectively. ρ represents the trade-off between classification accuracy and incremental learning ability whereas β is the factor by which the hyper-boxes are adjusted with each training pattern during the training phase. This approach is efficient for showing high accuracy, quick training time and ability for incremental learning.

2.4.3 Design of Classifier using Neuro Fuzzy Technique

Wang, Lee and Dillon, implemented generalized radial basis function (GRBF) neural network for extraction and optimization of fuzzy protein sequences clas-

sification rules [37]. A typical GRBF architecture is shown in Figure 2.10.

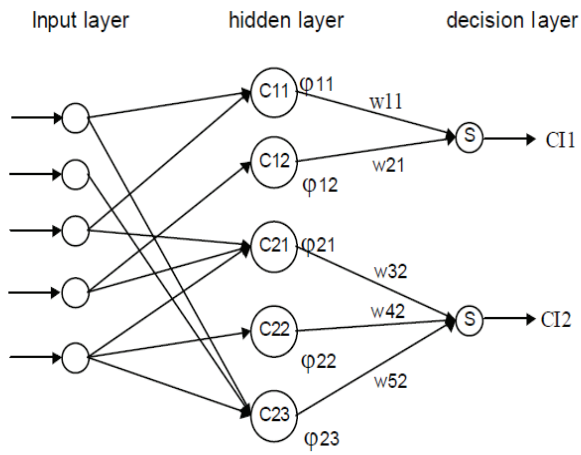

Figure 2.11: Architecture of GRBF Network

The network consists of 'm' input features $X = [x_1, x_2, \cdots, x_m]^T$, M hidden units and n output units in the decision layer. The activation function ϕ in the hidden units are the Gaussian functions defined by:

$$\phi(X_j) = exp[-d(X_j, C_j)] \tag{2.9}$$

where $X_j = [x_{j1}, x_{j2}, \cdots x_{jp}]^T$, $j_p \leq m$ represents a subset or a projection of X onto a subspace of the feature space, which is the contributory input vector to the j-th hidden unit. If C_j is the corresponding cluster center of the unit, $d(X_j, C_j)$ represents the weighted Euclidean distance measure. It is calculated as:

$$d(x_j, C_j) = \sum_{k=1}^{p} (x_{jk} - c_{jk})^2 / (\sigma_{jk})^2 \tag{2.10}$$

where (σ_{jk}) represents the variance of the Gaussian Kernel. A fuzzy T-norm operator, namely fuzzy plus operator $\oplus$ defined by:

$$a \oplus b = a + b - ab \tag{2.11}$$

is applied as the activation function at the output layer of the GRBF network.

2.4.4 Classifier using Support Vector Machine (SVM)

SVMs were developed by Cortes and Vapnik (1995) for supervised binary classification which is based on the well developed statistical learning theory. The non-linear-SVM maps the non-linearly distributed input data into a high dimensional feature space H by using kernel mapping function $\phi(x)$. SVM finds a hyperplane, which maximizes the margin, i.e., the distance between the hyperplane and the nearest data points of each class in the space H. The hyperplane can be described by $w.x + b = 0$ where:

- w is normal to the hyperplane.
- $\frac{b}{\|w\|}$ is the perpendicular distance from the hyperplane to the origin.

Support Vectors are the points closest to the separating hyperplane and the aim of SVM is to orientate this hyperplane in such a way as to be as far as possible from the closest members of both classes.

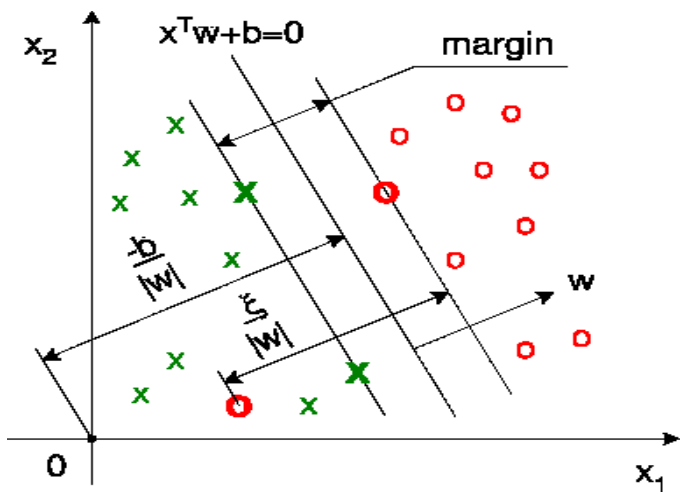

Figure 2.12: Representation of margin and support vectors in SVM

The mapping function $\phi(.)$ is implemented by a kernel function $K(x_i; x_j)$ which denotes an inner product in the space H. The most commonly used kernel function are as follows:

- Polynomial Kernel: $k(x, y) = (x.y)^d$

- RBF kernel : $k(x, y) = \exp(-\frac{\|(x-y)\|^2}{2\sigma^2})$
- Hyperbolic Tangent (Sigmoid Kernel) $K(x, y) = tanh(\alpha(x.y) + C)$

SVM classifier is implemented to classify protein sequences into corresponding families in [27] . From numerical simulation results it is observed that the technique is really effective for protein superfamily classification. Features selected based on relative entropy and SVM for classification is implemented in [23].

2.4.5 Classifier using Principal Component Null Space Analysis (PCNSA)

French et al. implemented PCNSA, a linear classifier for protein superfamily classification in [38]. In the first step, principal component analysis (PCA) was used on the entire training set for dimension reduction. In the second step, a null space for each class was found, which was extracted by taking the dimensions with the least variance of each class using eigenvalue decomposition technique.

2.4.6 Classifier using Nearest Neighbour Rule

The nearest neighbour method is one of the simplest classifier to predict the class membership of unknown test sample. This method is based on the distance metric between testing and training samples. The basic concept of this approach is based on the distance measure i.e., one training sample 't' is found for each test sample 's', with most similar expression value. The distance metric can be any similarity measure based on attribute values, e.g., the Pearson's correlation coefficient, the Euclidean distance function, etc.

Let E(x) represents expected value of vector x.
Var(x) represents variance of vector x.

s: represents test sample vector.

t: represents training sample vector.

The Pearson's correlation coefficient can be calculated as:

$$P(s,t) = \frac{E((s_i - E(s))(t_i - E(t)))}{\sqrt{var(s)var(t)}} \tag{2.12}$$

The class label of 't' is assigned to 's' by the nearest neighbour classifier expressed as :

$class(t,s) = class(argmax_i\ P(s,t_i))$ where class returns the class of training sample that has highest P value.

S. Bandyopadhyay implemented nearest neighbour rule to classify an unknown protein sequence into a particular superfamily based on the proximity to the prototype evolved using the genetic fuzzy clustering technique [25]. The time requirement as shown by the author is significantly less as compared to BLAST as well as shows better performance in classification.

2.5 Parameters used for Measuring the Efficiency of Classifier

For any classification problem, the outcomes of the data are always labelled i.e 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 as true positive. 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 it is referred as false negative. The measures used by most of the researchers to evaluate the performance of classifier are:

1. Precision = $[\frac{TP+TN}{TP+FP+TN+FN}] * 100\%$

2. Sensitivity (or True Positive Rate) = $\frac{TP}{TP+FN} * 100\%$

3. Specificity (or True Negative Rate) = $\frac{TN}{TN+FP} * 100\%$

4. Unclassified$_p$ = $\frac{N_{up}}{N_{po}} * 100\%$

5. Unclassified$_n$ = $\frac{N_{un}}{N_{ng}} * 100\%$

   where TP = number of true positive samples

   TN = number of true negative samples

   FP = number of false positive samples

   FN = number of false negative samples

   N_{up} = total number of positive test sequences

   N_{un} = total number of negative test sequences

   N_{ng} = total number of negative test sequences

   N_{po} = total number of positive test sequences

6. Receiver Operating Characteristic (ROC) Curve:

   ROC analysis investigates and employs the relationship between sensitivity and specificity of a binary classifier. Sensitivity or true positive rate measures the proportion of positives correctly classified; specificity or true negative rate measures the proportion of negatives correctly classified. The best possible prediction method would yield a point in the upper left corner or coordinate (0, 1) of the ROC space, representing 100% sensitivity (no false negatives) and 100% specificity (no false positives). The (0,1) point is also called a perfect classification.

7. Mean Square Error (MSE) and Number of Epochs: The number of epochs is the successive number of iteration, the neural network undergoes to meet the convergence criteria. The convergence criteria is fixed by assigning a threshold value to MSE. MSE is defined as :

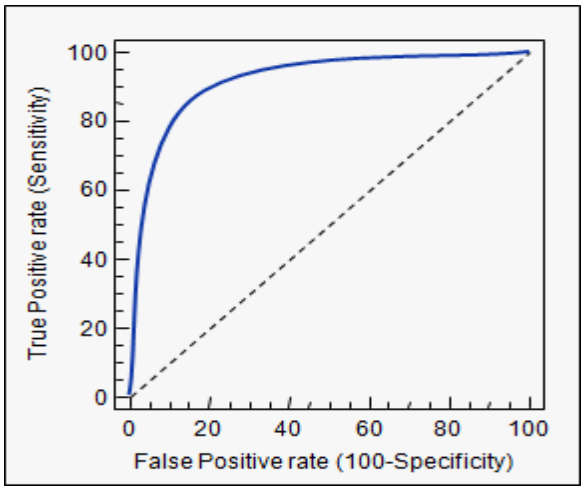

Figure 2.13: Representation of ROC curve

$$MSE = \frac{1}{2}\sum_{k=1}^{n}\sum_{i=1}^{q}[t_i(k) - o_i(k)]^2 \tag{2.13}$$

The above equation gives the vectorial difference between the k-th target output vector t(k) and the k-th actual vector o(k) of the network. 'n' denotes the number of training patterns presented to the network for learning purposes and 'q' denotes the number of nodes in the output layer. The learning of neural network stops when the MSE value falls below the pre-specified threshold value.

8. CPU Execution Time: The actual time required by the algorithm to meet the convergence criteria.

9. Mean, Variance and Standard Deviation: To measure the performance of randomized algorithms, the standard deviation is calculated over n number of observations. This can be calculated as :

$$\mu_x = \bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{2.14}$$

where μ_x denotes the mean over 'n' number of observations. To obtain the measure of the variability of the data, the statistics most often

used are the standard deviation $\sigma_x = \sqrt{(\sigma_x)^2}$ and variance $(\sigma_x)^2$. The standard deviation can be calculated as:

$$(\sigma_x) = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \mu_x)^2}{n}} \tag{2.15}$$

2.6 Conclusion

The application of various computational intelligence techniques applied by earlier researchers to the problem of protein superfamily classification is discussed in this chapter. The review has been done under two broad parts, such as feature extraction and selection, and design of the classifier. The various parameters used to measure the performance of the classifier are also discussed. The various approaches for global and local feature selection from amino acid sequence; as well as the dimension reduction techniques for feature extraction are discussed. Efficient design of the classifier is a key issue for any classification problem. Various soft computing techniques already implemented by the researchers for the design of classifier are reviewed. The results obtained from numerical simulations of some existing techniques are shown. It can be concluded that, the intelligent techniques possess the real challenge to handle the biological data as they possess the ability to exploit the tolerance for imprecision, uncertainty and partial truth thereby achieving tractability, robustness and low solution cost. Biological data are typically very large, complex, prone to noise, and change with time. The computational intelligent techniques offers a promising solution to handle and manipulate these type of data.

In the next chapter, the proposed method for Protein Superfamily Classification using Adaptive Evolutionary Radial Basis Function Network is discussed.

Chapter 3

Protein Superfamily Classification using Adaptive Evolutionary Radial Basis Function Network

---

In this chapter, the concept of Adaptive Multiobjective Genetic Algorithm (AMOGA) is applied for the structure optimization of radial basis function network (RBFN).

The modification to the earlier approach of Multiobjective Genetic Algorithm (MOGA) is done based on the two key controlling parameters such as *probability of crossover and probability of mutation*. These values are adaptively varied based on the performance of the algorithm i.e. based on the percentage of total population present in the best non domination level. PCA is used for dimension reduction and significant features are extracted from long feature vector of amino acid sequences.

3.1 Introduction

The problem of protein superfamily classification can be mapped as a pattern classification problem. For any pattern classification, the first step is retrieval of input patterns from any publicly available database, or from any reliable source. In case of retrieving biological data, UNIPROT, PIR, NCBI etc. databases are referred for retrieving gene and protein data. The data so derived may be incomplete, noisy (containing errors, or outlier values that deviate from the expected), and inconsistent (e.g., containing discrepancies). So after the inputs are retrieved, the next step is *data pre-processing*, where the data having missing values are filled following certain techniques. In this step, smoothing of noisy data are done; outliers are identified and removed; and inconsistencies are resolved.

In *Feature measurement* the dimension or number of attributes of every sample is measured. The number of samples collected from every class are also taken into account. The next step is *feature selection*, where subset of distinguishing features are selected from the original feature set as they have a high impact on the performance accuracy of the classifier. In feature extraction, 'D' dimension feature vector is reduced to 'm' ($m \ll D$) dimension using some linear or non-linear transformation techniques.

After the reduced feature vector is obtained, the entire data set is divided into training and test set. The training set is used for the learning (or training) of the classifier in which the classifier learns and gets adapted to the training patterns. The neural networks undergoes *supervised learning*, where every data in the training pattern are associated with a class label or target. The learning continues with an objective to reduce sum of costs for the training patterns. Once trained, the efficiency of the classifier is measured on the test data set (untrained patterns) in terms of generalization error or performance accuracy.

3.2 Basic 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. The inputs of hidden layer are the linear combinations of scalar weights and the input vector $[x_1, x_2, \cdots x_n]^T$ 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 $[y_1, y_2, \cdots y_m]^T$ for m outputs by linear combination of the outputs of the hidden nodes to produce the final output.

$$y = f(x) = \sum_{i=1}^{k} w_i \phi_i(x) \tag{3.1}$$

where f(x) is the final output, $\phi_i(x)$ 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. The architecture of RBFN is shown in Figure 3.1. A normalized Gaussian function is usually used as the radial basis function, that is

$$\phi_i(x) = \left(-\frac{\| x - c_i \|^2}{2(\sigma_i)^2} \right) \tag{3.2}$$

where $[x_1, x_2, \cdots x_n]^T$ denotes the input vector, $[c_1, c_2, \cdots c_m]^T$ denotes the ith center vector and $(\sigma_i)^2$ represents the width parameter of the radial basis function.

RBFN are an effective tool for pattern classification problem as they have good generalization and approximation ability with a simple network structure. In generalized RBF, *the supervised learning* of the center 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 [39].

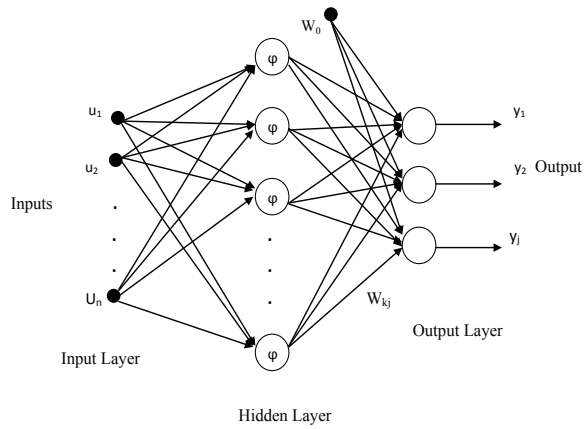

Figure 3.1: Architecture of Radial Basis Function Network

3.2.1 RBFN as an efficient classifier

Neural networks are efficient tool for pattern classification. From the implementation of FFNN trained using BP algorithm, PNN and RBFN, it is observed that PNN had outperformed the other two networks (as shown in Figure 2.9). From simulations it was observed that, as the size of the training set increases, the FFNN trained using BP algorithm, takes too long time to converge so as to reach at a predefined MSE value. As BP algorithm is an iterative process, the functional signals flows in forward direction and error signals propagate in backward direction to update the connecting synaptic weights. The two major drawbacks of BP algorithm are: very slow computing speed and the possibility of getting trapped in local minima. Although PNN performed well, but, the major drawback of PNN is that, it performs well when the training data set is small in size. But as the size of training data set increases, the architecture of PNN becomes very large and complex. This is because every node in the pattern layer represents a training sample. Practically it is observed that, there is an exponential growth in size of protein database as many newly discovered protein samples are added into the database. So, PNN cannot be the right choice for an efficient classifier to solve the practical problems. For these reasons, RBFN is preferred to perform classification task,

to classify amino acid sequences to their superfamily.

RBFN randomly selects subset of training samples to form the nodes of the hidden layer which contradicts the concept of formation of nodes of the pattern layer in PNN. The hidden layer controls the complexity and generalization ability of the network. These hidden centers are often randomly chosen as subset from training data points or K-means clustering algorithm may be used to cluster data points where every cluster center represents a node in the hidden layer. The problem of finding the optimal number of hidden centres remains a critical issue in the design of RBFN. Many evolutionary approaches are suggested to optimize the structure of RBFN. A modification to the earlier approach of MOGA is implemented here, which adaptively manipulates the probabilities of crossover and mutation based on the number of solutions present in the best non-domination level. The main objective is to derive the optimal structure of RBFN from the pareto optimal set and then apply it for protein superfamily classification problem. The effectiveness of the two approach i.e., MOGA and AMOGA are measured in terms of accuracy and convergence rate.

3.3 Related Work on Structure Optimization of RBFN using Evolutionary Techniques

Many evolutionary optimization techniques were successfully implemented for structure optimization of RBFN. Yen and Liu implemented Hierarchical Rank Density Genetic Algorithm (HRDGA) to evolve the neural network topology and parameters [40]. The rank-density based fitness assignment technique was used to optimize the performance and topology of the evolved neural network to solve the two conflicting multi objectives such as training performance and network complexity. Oliver Buchtala et al. used an evolutionary algorithm (EA) that performed feature and model selection simultaneously for RBFN. It

was validated in the area of intrusion detection in computer networks, biometric signature verification, customer acquisition with direct marketing methods, and optimization of chemical production processes [41]. Differential Evolution (DE) algorithm, a new promising evolutionary technique was proposed to train RBFN related to automatic configuration of network architecture. Classification tasks on data sets such as: Iris, Wine, New-thyroid, and Glass were conducted to measure the performance of neural networks [42]. Multiobjective Optimization (MOO) using rank method such as Fonsecas ranking, was implemented in [43], to optimize the structure of RBFN and thereby solving the trade-off between architecture and performance of classifier. Ensemble of RBF networks was obtained using the evolutionary multi-objective optimization method [44]. In this method, the RBF network structure was encoded in the NSGA-II chromosome based on two evaluation criteria, i.e. the accuracy and complexity of the model. Particle Swarm optimization (PSO) based multiobjective training was implemented in [45] for simultaneous optimization of architectures and connection weights.

Implementation of GA for structure optimization of RBFN is shown in [46] 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. LinGuo et al. used GA to optimize the parameters of RBFN and a hybrid learning algorithm further adjusts the parameter values [47]. J.Gonzalez et al. implemented Multiobjective evolutionary algorithm (MOEA) in which global mutation operators based on matrix transformation such as SVD and orthogonal least square (OLS)have been used [48]. Multiobjective structure selection method using MOGA is shown in [49], 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. GA with hybrid learning algorithm (HLA) have outperformed GA, ROLSA and K-means clustering with HLA. HLA mostly adjusts the centres and the widths [50]. 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 [51]. A two level learning method for designing an optimal RBFN using adaptive velocity update relaxation PSO (AVURPSO) and OLS is implemented in [52].

3.4 Brief Overview of the Entire Process

Choosing an appropriate set of relevant features is a very critical task for any classification problem. Too many features may include redundant and noisy values which may increase the computational complexity of the classifier. Similarly, too less number of features may reduce the generalization ability of the classifier. Therefore, selecting optimal number of distinguishing features is highly necessary for maintaining a high level of performance accuracy of the classifier.

Feature selection selects a subset of finite number of features but feature extraction creates new feature based on transformation of the original feature set using some dimension reduction techniques. Here, bi-gram measure is used for feature selection which in turn gives rise to large redundant sparse matrix. To reduce the dimension, PCA is used, which derives significant patterns by rotating the feature vector across the highest variance principal components derived from the covariance matrix. RBFN obtained from the pareto optimal set of MOGA and AMOGA are used as classifier. To evaluate the performance of two approaches, *convergence rate and predictive accuracy* of the classifier are taken into account. Gaussian spread (σ) of the radial basis function is the controlling parameter and the algorithm was run many times by varying various (σ) values. The overall process of the experiment is shown in Figure 3.2.

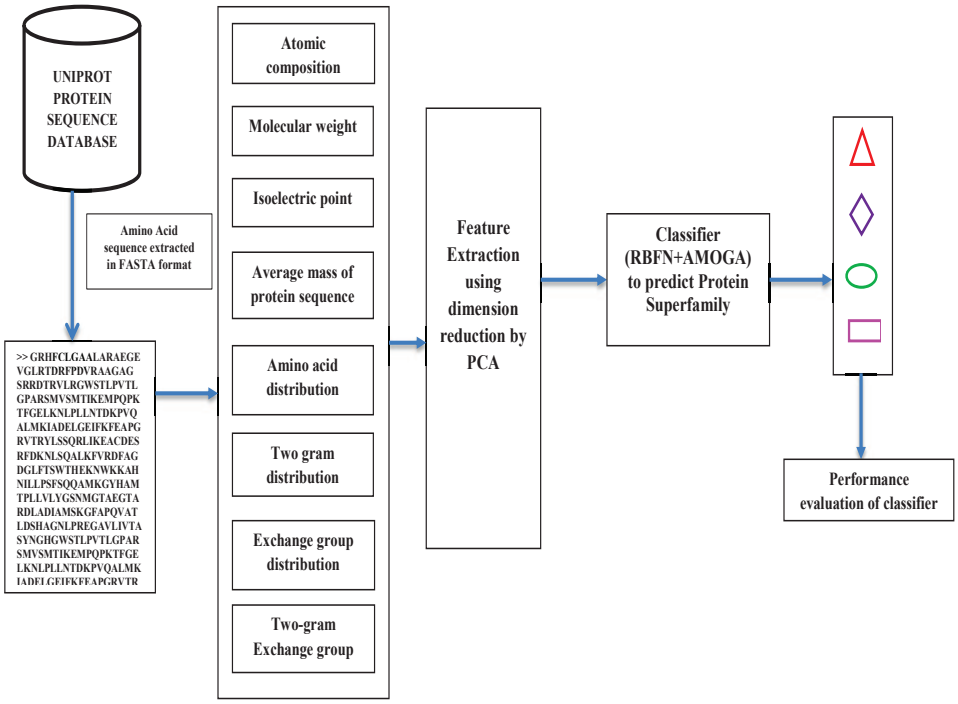

Figure 3.2: Brief overview of the entire process

3.5 Feature Extraction from Amino Acid Sequence

In general, the genetic code specifies 20 standard amino acids (described in Annexure I) such as:

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

The schematic representation of feature extraction is shown in Figure 3.3.

For protein feature selection, the two gram features such as [(AA, AC $\cdots$ AY), (CA, CC $\cdots$ CY), $\cdots$ (YA, YC $\cdots$ YY)] 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 [17]. Apart from this, bi-grams reflecting the pattern of substitution of amino acids

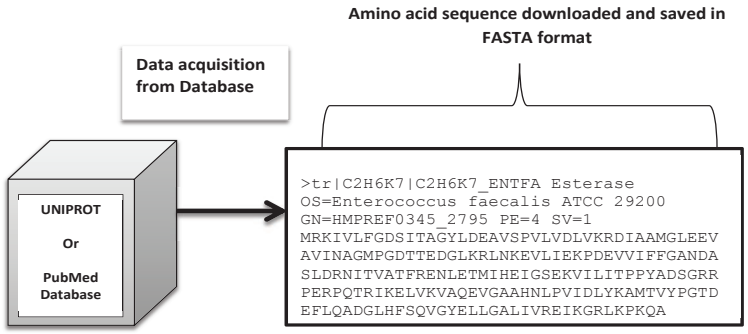

Figure 3.3: Schematic representation of feature extraction from amino acid sequence

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) [16]. Exchange grams are similar but are based on a many to one translation of the amino acid alphabet into a six letter alphabet that represents six groups of amino acids, which represent high evolutionary similarity. Generally the exchange groups used are:

$$e1 = \{H, R, K\},\ e2 = \{D, E, N, Q\},\ e3 = \{C\}$$

$$e4 = \{S, T, P, A, G\},\ e5 = \{M, I, L, V\},\ e6 = \{F, Y, W\}$$

The exchange groups statistically describes the probability of one amino acid replacing another over time. The total number of possible bi-grams on these six substitution groups is 6^2, that is 36. Thus, the overall bi-gram features extracted computes 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), exchange group distribution (6) and some other features are also taken into account.

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

$X^{(1)}, X^{(2)}, \cdots, X^{(5)}$ = atomic composition

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

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

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

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

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

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

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

If total 'n' number of instances is assumed, then the matrix size becomes n×470 which is a matrix having large number of sparse entries. PCA, a very powerful statistical technique for dimension reduction is used to retrieve significant patterns by projecting data into lower dimension. The projection is basically done by selecting the eigen vectors (or PC's) from covariance matrix showing cumulative variance upto level of 99%. A sample of input matrix is shown in (Annexure II).

3.6 PCA for Dimension Reduction

The concept of PCA was developed by Karl Pearson in 1901. 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 [53]. The implementation of PCA for feature extraction is implemented in [30].

First, the covariance matrix S is computed and eigenvalues are found. The eigenvalues are sorted in a decreasing order and let they are denoted as $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_M$. Let the corresponding eigen vectors be denoted as $a_1, a_2, \cdots a_M$.

The first d eigen vectors are selected from M vectors such as $d \ll M$. Finally, the data set is projected into lower dimension as given by:

$$G \leftarrow [a_1 a_2, \cdots a_d] \text{ where } d \ll M.$$

if x is a test point

$$x \epsilon R^M \rightarrow x\ G\ \varepsilon\ R^d \tag{3.3}$$

The detailed steps of PCA is described in Algorithm 1.

Algorithm 1 $PCA(\ )$

Algo: PCA for dimension reduction.

Input: n×m feature matrix X where n represents number of samples and m represents the number of features.

Output: n×k reduced feature matrix ($k << m$).

Step 1: Normalize the matrix X to ensure zero mean of each feature value.

Let training set $= x^1, x^2 \cdots x^m$

Evaluate $\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_i^j$ vary j for all feature values i.e 1 to m

Replace x^j with $(x^j - \mu_j)$ vary x^j across all samples i.e from 1 to n

Step 2: Compute covariance matrix of the normalized matrix.

$\sum(sigma) = \frac{1}{m}(X^T X)$

Step 3: Compute the eigen vectors of matrix using MATLAB command as: $eign = eig(sigma)$

Step 4: Choose the first k number of principal components from the covariance matrix using the following criteria:

for (*every eigen vector* $i = 1$ *to* m) **do**

Evaluate $cumvar = \frac{\sum_{i=1}^{k} \lambda_{ii}}{\sum_{i=1}^{m} \lambda_{ii}}$

{cumvar denotes cumulative variance and λ represents eigen values sorted in descending order.}

if $(cumvar \geq 0.99) or (1 - cumvar \leq 0.01)$ **then**

return k {99% of variance is retained.}

end if

end for

Step 5: Reduce the matrix dimension, taking the first k columns (1 to k) of eign matrix as eign(:,1:k) and assign to $eign_{red}$.

Step 6: Evaluate $Z = X * eign_{red}$ where Z is the new matrix with reduced feature dimension retaining 99% of the variance.

STOP.

3.7 Multiobjective Optimization

In single objective optimization problem, there is one global optimal solution and the solution having higher level of information is chosen. But as most of the real world problems are complex, in the sense, they may be non-linear, multi modal and stochastic, there may be more than one parameters which need either to be minimized or maximized. These type of problems are referred to as multi objective optimization problem which can be solved by various approaches. A survey of various multi objective evolutionary techniques is described in [54]. The most simple method among all approaches is, to form a composite objective function as the weighted sum of various objectives and weight value is assigned as per the priority of individual objective. This technique is otherwise referred as *preference based multi objective optimization* in

which the multiobjective problem is mapped into a single objective problem which is a composite of more than one objective.

$$F = w_1 f_1 + w_2 + f_2 + \cdots + w_n f_n \tag{3.4}$$

The multi-objective evolutionary algorithms (MOEAs) can be broadly classified into two broad categories namely elitist and non-elitist. In multi-objective optimization problems there may exist some cases where the objectives are conflicting to each other. Generally, these problems can be solved by making a pair wise comparison and arranging them in several non domination fronts (or pareto fronts) based on their rank. The two main goals for any pareto optimal solutions are :

- The solution should converge as close as possible to the true pareto optimal front.
- The solutions should be as widely spread as possible on the best pareto front.

The crossover (or the recombination) and the mutation operator controls the evolution process by bringing diversity in the solution space. As evolution takes place, it is observed that the local pareto optimal fronts converges towards the global pareto optimal front. The schematic representation of pareto optimality and dominated points is shown in Figure 3.4.

Thus, a multi objective optimization problem (MOP) has a number of objective functions which are either to be minimized or maximized. The objective functions may be conflicting to each other and are subjected to some constraints.

3.8 Basic concept of Adaptive MOGA (AMOGA)

AMOGA strictly follows the concept of "survival of fittest" where very good solutions having high fitness values are well protected and the solutions with

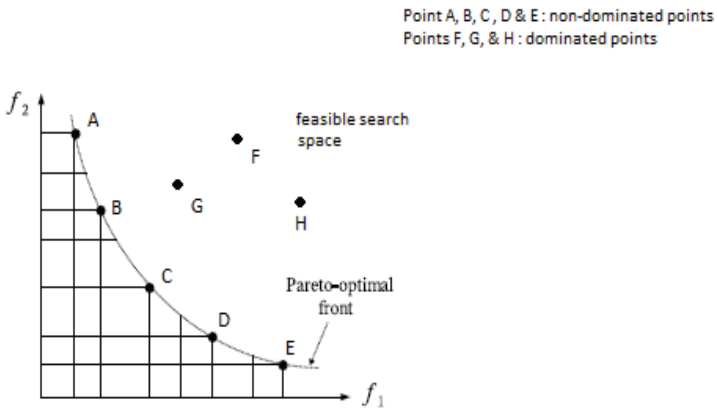

Figure 3.4: Schematic representation of pareto optimality and dominated points

poor fitness values are eliminated. The basic idea behind the implementation of AMOGA (with non-dominated sorting ($NSGA-II$) as suggested by Deb et al. in [55–57])is that, the probabilities of crossover and probabilities of mutation (P_c and P_m)are varied based on the number of solutions present in the best non domination level. Though, GA is a randomized search technique, still the search in MOGA progresses in the direction of convergence of solutions to the global pareto front. If the P_c and P_m values are kept constant, it may so happen that, as the generation progresses towards global optima, the good solutions may get disrupted and may move to other higher fronts by which the process may take too long time to converge. The exploration of search space and exploitation of non-dominated solutions are greatly controlled by the values of P_c and P_m. The number of solutions in the best non-domination level is the yardstick to adaptively control and manipulate the values of P_c and P_m. In other words, when the obtained pareto front merges towards the true global pareto front, P_c and P_m values are adaptively decreased to prevent disruption of very good solutions. The detailed steps of AMOGA are described in Algorithm 2.

Algorithm 2 *AMOGA*( )

Let population size = N

Probability of crossover = P_c

Probability of mutation = P_m

Let Fitness function be denoted as : $f_1, f_2 \cdots f_n$

Let Pareto fronts be denoted as : $F_1, F_2 \cdots F_m$

No. of solutions in the first pareto front = l

Step 1: initialize population P_0 ;

Step 2: evaluate fitness function based on objective functions;

Step 3: perform non dominated sorting and generate pareto optimal fronts;

Step 4: calculate the crowding distance of all solution points;

Step 5: select the best (N/2) solutions based on their fronts and crowding distance. Let these solutions denote the new parents P_t.

Step 6: perform tournament selection by selecting N random pairs from P_t. Use the crowded comparison operator ($\preceq_c$) to select the most widely spread solutions which are the winners of the tournament.

Step 7: perform pairwise crossover and bit wise mutation to create new offspring. Let the new population be denoted as P_{t+1}.

Step 8: now evaluate the fitness of new population P_{t+1}.

Step 9: let number of solutions in F_1 be denoted as $|F_1| = l$

Step 10:

if $(l \geq (n/N)\%)$\{check l with respect to n, 2n, 3n, $\cdots$\} **then**

$P_c = P_c - constant\ step\ factor$

$P_m = P_m - constant\ step\ factor$

end if

Step 11:

if $(l \geq (predefined)\%)$ {termination condition met} **then**

exit

else

goto step 2

end if

3.9 Structure Optimization of RBFN using AMOGA

The problem of finding the *number of hidden centres* remains a critical issue in the design of RBFN. The number of basis function controls the complexity and generalization ability of the network. If more number of training samples are selected as hidden centres, this may include redundant samples which results in large network structure. Thus, the computational overhead is too high when an unknown pattern is classified. The network will also have poor generalization capability as it becomes over sensitive to the training data and thereby recognizes the noisy samples as patterns. On the contrary, very few number of hidden centers in the hidden layer may lower the classification accuracy of the trained network. Thus, a trade-off between the accuracy and the computational complexity arises which can be solved by selecting optimal number of hidden centres from the pareto optimal set. Besides the number of hidden centers, the weight matrix connecting the hidden and output layer also affects the accuracy of the classifier.

In the implementation of RBFN-AMOGA, every chromosome has two parts. The first part is encoded as binary string which either selects or discards a sample for being the hidden center. The second part encodes numeric values which are converted to decimal values using weight extraction formula. As generation evolves, the pareto fronts generated using $NSGA-II$ can solve the trade-off problem for designing an optimal structure of RBFN showing good performance in terms of classification accuracy. The percentage of total number of solutions in the best non-domination level is the yardstick to manipulate the p_c and p_m values. The most optimal RBF network with good generalization ability can be derived from the pareto optimal set. Therefore, every solution of the pareto optimal set gives information regarding the specific samples to be chosen as hidden centers as well as the update weight matrix connecting the hidden and output layer.

3.10 Experiment Details and Simulation Results

3.10.1 Input Details:

The amino acid sequences are downloaded in FASTA format from UNIPROT repository. The four super-families considered for numerical simulations are Globin, Kinase, Ribitol dehydrogenase and Ligase from Uniprot repository. (`http://www.uniprot.org/`).

3.10.2 Details of using AMOGA

1. Initialization of chromosome: The population size was fixed at N=40.

   P_c=0.8

   P_m=0.008

   The genotype of the chromosome consists of two parts. The first part is binary encoded which controls the topology of the network by choosing the optimal number of relevant basis functions. The second part encodes the synaptic weight which gets optimized as the generation evolves, to improve the generalization ability of the network. The schematic representation of chromosome is shown in Figure 3.5.

   Here, each weight is represented as a five digit number and the weights are extracted using the following weight extraction formula:

   Let $g_1, g_2, ..g_d, ..g_l$ represent a chromosome where g(d) represent a gene of the chromosome. Let $g_{kd+1}, ..g_{kd+2}..g_{k+1d}$ represent the k-th gene ($k \geq 0$) in the chromosome. The actual weight W_k is given by :

$$W_k = \begin{cases} +\frac{g_{kd+2}10^{d-2}+g_{kd+3}10^{d-3}+\cdot g_{(k+1)d}}{10^{d-2}} & \text{if}(\ 5 \leq g_{kd+1} \leq 9) \\ -\frac{g_{kd+2}10^{d-2}+g_{kd+3}10^{d-3}+\cdot g_{(k+1)d}}{10^{d-2}} & \text{if}\ (0 \leq g_{kd+1} \leq 5). \end{cases}$$

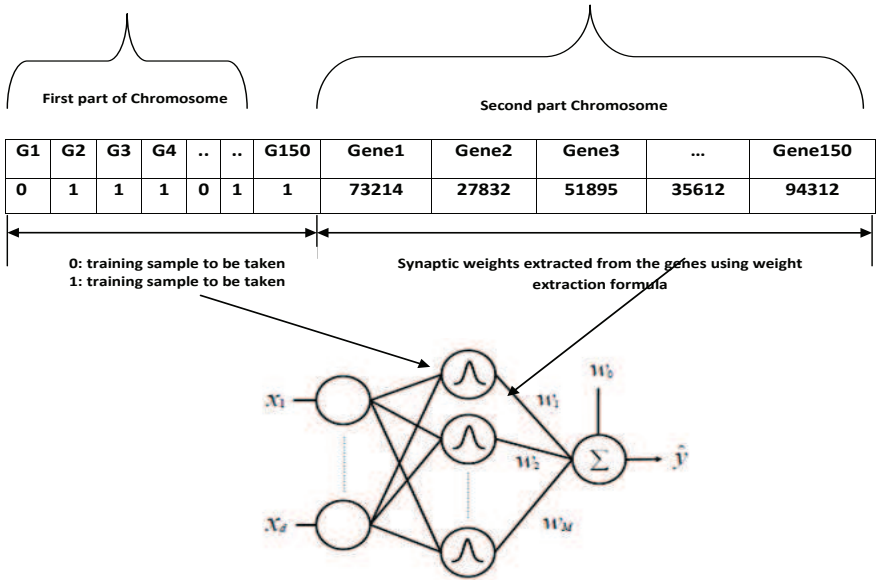

Figure 3.5: Representation of chromosome

The second part of chromosome has variable number of genes as it depends on the number of 1's present in the first part of chromosome.

2. Evaluation of fitness function: The first objective function
   (f_1): is to minimize the number of hidden centers.
   The second objective function:
   (f_2): is to minimize the MSE which is the difference between the neural network output and the target output.
   The fitness function may be defined as follows:

$$f2 = minimize(R) \tag{3.5}$$

where,

$$R = \frac{1}{2}\sum_{l=1}^{L}\sum_{j=1}^{J}(e_j)^2(n) \tag{3.6}$$

where J is the total number of neurons in output layer, L is the number of training samples, $e_j(n)$ represents the error signal which is the difference between desired output d and the output obtained.

3. Assignment of rank based on fitness values: Based on two objective functions, there are two fitness values f_1 and f_2 for every chromosome. The non-dominated sorting $(NSGA - II)$ gave a rank to every solution and based on rank values many non dominated fronts were obtained.

4. Crowding distance assignment and Binary tournament selection:

   Crowding distance (CD) was assigned to every solution and few randomly chosen (N/2) number of individuals from the best non domination levels were sent to the mating pool. The crowding distance values guides to select subset of solutions from a pareto front so as to fill (N/2) of the population size.

   Crowding distance estimation: The boundary solutions lying on the pareto front are assigned infinity values. The crowding distance for intermediate solutions in the pareto are estimated as :

$$d_{Ij}^{m} = d_{Ij}^{m} + \frac{f_m^{(I_{j+1}^m)} - f_m^{(I_{j-1}^m)}}{f_m^{max} - f_m^{min}} \tag{3.7}$$

   The index I_j denotes the solution index of the jth member in the sorted list. Thus for any objective, I_1 and I_l denotes the lowest and highest objective function values respectively, which are assigned to infinity. $f_m^{(I_{j+1}^m)}$ and $f_m^{(I_{j-1}^m)}$ denotes objective function values between two neighbouring solutions on either side of solution I_j. f_m^{max} and f_m^{min} are the maximum and minimum population values for the mth objective function. N number of random pairs were selected for the tournament and the winner from the two individuals were decided based on the crowded comparison operator $(\preceq_c)$. The crowded comparison operator $(\preceq_c)$ guides the selection process at the various stage of the algorithm towards a uniformly spread out pareto optimal front.

   case 1: $i \leq j$ i.e., solution i has a better rank, if $(i_{rank} < j_{rank})$

   or

case 2: if ($i_{rank} = j_{rank}$) then ($i_{distance} > j_{distance}$)
where i_{rank} shows non-domination rank and $i_{distance}$ is the crowding distance of the $i - th$ individual. The first condition selects individual on better non dominated front where as the second condition resolves the tie by choosing the solution having higher crowding distance.

5. Adaptive crossover and mutation:

   The P_c and P_m values were adaptively varied when the intermediate criteria were met. While performing the simulation, the following assumptions were made:

   if ($|F_1| \geq 25\%$)
   {
   $P_c = P_c - 0.2;$ $\qquad (n = 25\%,\ 2n = 50\% \cdots)$
   $P_m = P_m - 0.002;$
   }
   The probability values were updated when $|F_1|$ was more than 50% , 75% and 90% respectively.

6. Stopping criteria: The AMOGA process terminates when 90% or more number of solutions are in the best non domination level.

3.10.3 RBFN Details:

The number of nodes in the input layer was decided on the basis of reduced feature vector dimension obtained after the implementation of PCA. The number of hidden nodes were selected from the pareto optimal front and the target vector constituted the output nodes in the output layer which is required for the evaluation of MSE.

Target vector for the four protein superfamilies are as follows:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.10.4 Parameters used for Measuring the Efficiency of Classifier:

The parameters used for measuring the efficiency of two approaches are convergence rate and predictive accuracy.

The concept of TP, TN, FP, FN are described in section 2.4.

$$\text{Predictive accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$$

Where TP=true positive TN=true negative FP=false positive FN=false negative.

3.11 Results and Discussion

After PCA was applied for significant feature extraction, 99% of the variance was retrieved by the first 57 principal components. The first ten PC's extracted are shown below in Table 3.1. The top 57 PC's were projected to map the original feature matrix to lower dimension. The reduced feature vector are given as input to RBFN and then implementation of MOGA and AMOGA were carried out.

The graphs obtained from numerical simulations, shows the better performance of AMOGA over MOGA for protein super family classification problem. So, if the probability values are kept constant, the better fitness solutions get disrupted and it may take longer time to converge. But varying the probabilities adaptively, the good solutions of the feasible search space are protected

Table 3.1: Variances and cumulative variances across first ten PCs

Principal Components	% of variances.	Cumulative % of Variance
PC1	66.9942	66.9942
PC2	5.2062	71.2004
PC3	4.0826	75.2830
PC4	3.6389	78.9219
PC5	3.4684	82.3903
PC6	2.8422	85.2325
PC7	2.6957	87.9282
PC8	2.3590	90.2872
PC9	2.0720	92.3592
PC10	1.6404	93.9996

and the algorithm converges faster to the global optima. For various ensembles of RBFN derived from pareto front, the comparative results shows the effectiveness of AMOGA over MOGA. The algorithms were run several times by varying various values of the Gaussian spread (σ) and it was observed that good ensembles of RBFN were derived when ($\sigma = 0.5$). The pareto fronts obtained after implementing AMOGA and MOGA by varying the σ values (σ = 0.3, 0.5 and 0.7) are shown in Figure 3.6, 3.7 and 3.8 respectively.

Tables 3.2 and 3.3 shows few solutions from the pareto optimal set obtained from the top, mid and lower region of the pareto front of MOGA and AMOGA (when $\sigma = 0.5$) respectively. MOGA converged after 1019 generations whereas AMOGA converged after 338 generations to meet the same stopping criteria. It can be concluded that, the adaptive nature of MOGA (AMOGA) helps in faster convergence of the evolution process. The number of correctly classified samples from individual superfamily are shown in the Tables 3.4 and 3.5. From the pareto front generated after simulations, it was observed that, when very

few centers are selected, misclassification error was high where as when more number of centers are selected, the misclassification error was low. The optimal number of centers were selected from the mid region of the pareto front which solves the trade-off problem between network complexity and accuracy. The specific centers to be selected in the design of RBFN were obtained from the first part of chromosome and the final updated output weight matrix were obtained from the second part of the chromosome.

To show the efficiency of RBFN-AMOGA over standard neural networks, the comparison process is further extended by implementing FFNN (trained using BP algorithm), PNN and Standard RBFN (trained using supervised gradient descent learning algorithm [39]). The learning rate (η) and momentum (α) are the two main controlling parameters of BP algorithm. Keeping $\alpha = 0.3$ fixed and varying (η) in the range of 0.1 to 1, variation in performance accuracies were observed. The smoothing parameter (σ) is the controlling parameter for PNN and RBFN. After deriving a particular 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 neural networks are shown in Table 3.6 and graph (Figure 3.9).

Table 3.2: Pareto optimal subset of first pareto front of MOGA after 1019 Gens.

Sl.No.	F1 (No. of centers)	F2 (Misclassification error)
1	48	0.0109
2	57	0.0093
3	59	0.0088
**4	62	0.0079
5	67	0.0067
6	76	0.0019
7	81	0.0013

Table 3.3: Pareto optimal subset of first pareto front of AMOGA after 338 Gens.

Sl.No.	F1 (No. of centers)	F2 (Misclassification error)
1	46	0.0109
2	55	0.0106
3	59	0.009
4	60	0.008
** 5	61	0.0062
6	68	0.0025
7	77	0.001

(** RBFN ensemble chosen from pareto optimal set for protein superfamily classificn.)

Table 3.4: Performance of MOGA

σ values	No. of Gens.	Pred. Accr. in %
0.3	787	94.79
0.5	1019	96.18
0.6	967	93.87

Table 3.5: Performance of AMOGA

σ values	No. of Gens.	Pred. Accr. in %
0.3	355	97.57
0.5	338	97.91
0.6	460	95.6

Table 3.6: Maximum performance accuracy achieved by neural networks

Sl.No.	Neural Networks	F2 Performance accuracy (in %)
1	FFN-BP	85.33
2	PNN	92.67
3	Standard-RBFN	84.13
4	RBFN-MOGA	96.18
5	RBFN-AMOGA	97.91

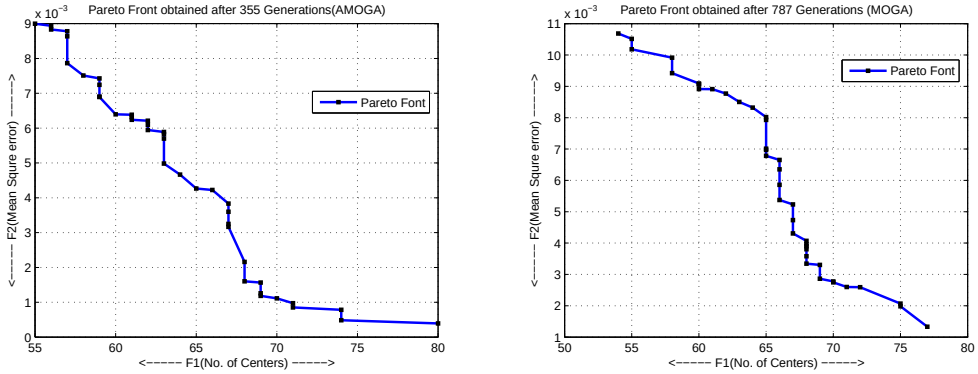

Figure 3.6: Performance of AMOGA and MOGA when $\sigma = 0.3$

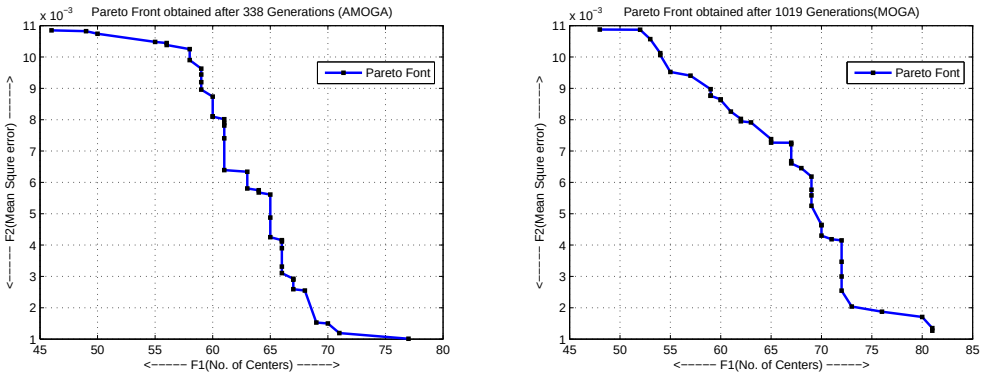

Figure 3.7: Performance of AMOGA and MOGA when $\sigma = 0.5$

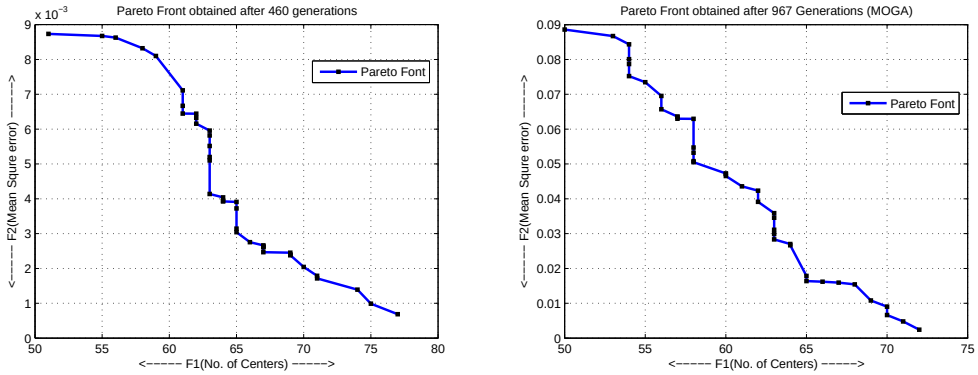

Figure 3.8: Performance of AMOGA and MOGA when $\sigma = 0.7$

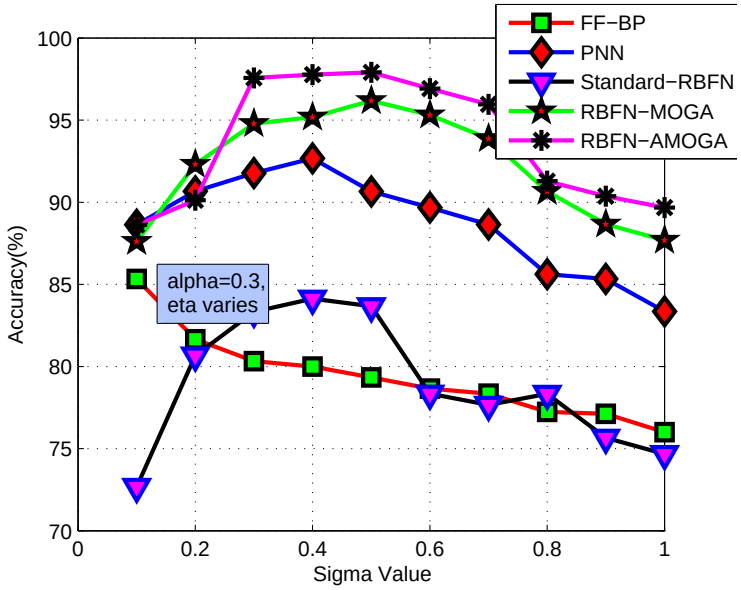

Figure 3.9: Performance accuracy of neural networks by varying the control parameters

3.12 Conclusion

From the numerical simulations, it can be concluded that AMOGA outperformed MOGA in terms of speed and accuracy by protecting the high fitness solutions from getting disrupted in the search space. Although, many evolutionary approaches are already suggested for optimizing the structure of RBFN, but the approach of AMOGA has shown faster convergence to the global pareto front thereby giving the optimized structure of RBFN. The RBFN obtained from the pareto optimal set has shown good classification accuracy in comparison to standard neural networks, which was validated by performing classification considering four protein superfamilies. The results obtained from RBFN-AMOGA are quite promising and this technique can be implemented by drug analyst and researchers to correctly classify protein to their superfamily.

In the next chapter, the proposed method of Two Stage Approach for Protein Superfamily Classification is discussed.

Chapter 4

Two Stage Approach for Protein Superfamily Classification

In this chapter, protein superfamily classification is done in two stages. In the first stage, optimal number of features are extracted using PCA-NSGA-II (non-dominated sorting GA) and in the second stage, Recursive Orthogonal Least Square Algorithm (ROLSA) is used to train RBFN.

4.1 Introduction

In previous chapter, although the implementation of AMOGA gave the optimized structure of RBFN showing a good level of performance accuracy, but further improvement to the technique was done to derive the most parsimonious structure of RBFN. Besides that, improving the classification accuracy of the classifier is one of the primary objective of any classification problem. RBFN-ROLSA is implemented for deriving the reduced structure of RBFN thereby maintaining a good level of performance accuracy. The traditional PCA algorithm selects the top few eigen vectors having large eigenvalues for dimension reduction. But this method of selection of eigen vectors might not

be the best choice always, as illustrated by Balci et al. [58] and Sun et al. [59]. The application of GA may guide to select a subset of eigenvectors encoding important information about the target concept of interest. The efficiency of GA-PCA approach is illustrated in [59] on two challenging applications such as vehicle detection and face detection.

In this proposed work, the classification problem is solved in two major stages. In the first stage, PCA-NSGA-II is implemented. This is a hybridized approach which tries to solve the trade-off problem between selection of optimal number of significant eigen vectors and 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, eigen vectors having non-zero eigen values are encoded in the chromosome. GA helps in searching the eigen space to select the distinguishing eigen vectors. PNN is used as inductive algorithm and the evaluation function used in this wrapper approach is, the minimization of the misclassification rate of PNN over the test samples. A detailed description of wrapper approach and inductive algorithm is discussed in [14]. The implementation of PCA-NSGA-II derives the optimal number of significant eigen vectors to build the reduced feature matrix. After deriving the reduced feature matrix, ROLSA is implemented for efficient design of RBFN in the second stage.

4.2 Feature Selection and Feature Extraction

Feature selection (also known as subset selection) is a process commonly used in machine learning, where a subset of features are selected that lead to the smallest classification error. The best subset contains the least number of dimensions that mostly contribute to the target concept of interest. Langley grouped different feature selection methods into two broad groups i.e., filter and wrapper approach. This categorization is based on the dependence on the inductive algorithm that finally uses the selected subset [60]. Filter methods

are independent of the inductive algorithm, where as wrapper methods use the inductive algorithm as the evaluation function.

The five main types of evaluation functions as suggested by [M. Dash and H.Liu in [14]] are:

- distance measure (euclidean distance measure).
- information content (entropy, information gain, etc.)
- dependency measure (correlation coefficient).
- consistency measure (min-features bias).
- classifier error rate (the classifier themselves).

The first four are the evaluation functions for the filter approach and the last measure is for wrapper approach.

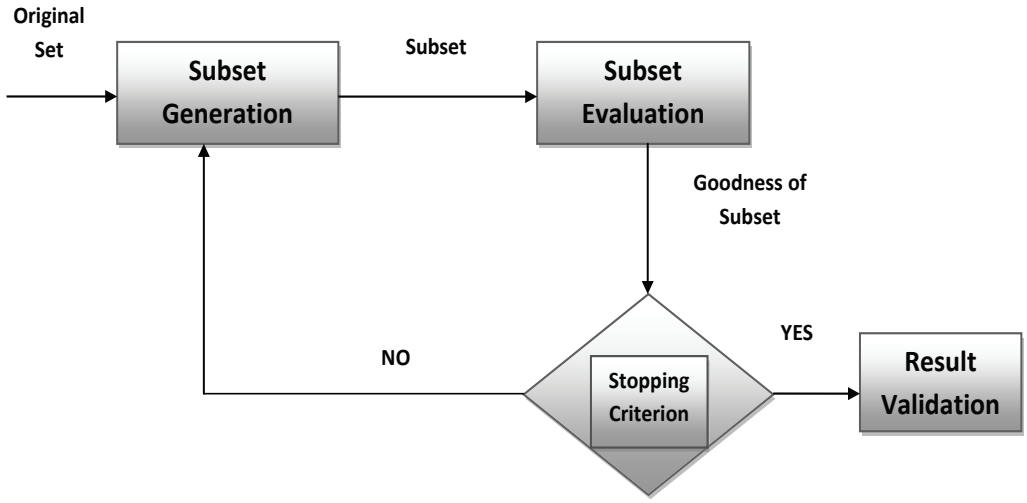

Figure 4.1: Feature selection process with validation

Anil Jain et al. had clearly distinguished the concept of feature selection from feature extraction [12]. 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 technique which performs some transformation of the original feature matrix to create new feature matrix by discarding the features having low discrimination ability.

4.3 Feature Extraction using PCA-NSGA II

Generally, traditional PCA selects the top few eigen vectors having higher eigen values. But eigen vectors having large eigen values, may not always have a great impact regarding the target concept of interest. This is validated in face detection and vehicle detection experiments that, the eigen vectors having low eigen values encode more lighting information and also encodes some specific local features [59]. GA is implemented which searches the eigen space to select subset of eigen vectors. The two objectives such as selection of minimum number of significant eigen vectors and minimization of classification error rate are solved using weighted sum approach by giving proper weight values to the objective functions. Zhao et al. 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 [27]. Zhao et al. developed a hybrid GA/RBFNN technique which selects features from protein sequences and train 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 [26].

In PCA-NSGA-II approach, PCA is hybridized with the elitist non-dominated sorting GA or NSGA-II for feature subset selection from the eigen space. The encoding of chromosome is done as string of 0's and 1's. The trade-off between the two objectives such as minimization of number of eigen vectors selected and minimization of misclassification error rate are solved by generating pareto fronts at various non-domination levels. The goodness of a chromosome is evaluated based on the number of 1's in the chromosome string. In the implementation, GA searches the eigen space comprising of top 63 eigen

vectors having non-zero eigen values. The eigen vectors above 63 have very small eigen values nearly equal to 0. So, the length of chromosome is fixed at 63. The transformation matrix is built on the basis of position of 1's in the chromosome. The original feature space is mapped to lower dimension matrix using the transformation matrix. The reduced feature matrix is then given as input to the PNN. PNN evaluates the second fitness value of chromosome i.e the misclassification error rate obtained by the selected eigen vectors, over the test sample.

PNN is the inductive algorithm which is wrapped with every chromosome for the evaluation of second fitness value. The best solutions are derived from the lowest level pareto front. The algorithm for implementing PCA-NSGA-II is described in Algorithm 3 and the brief overview of the entire process is shown in Figure. 4.2. The NSGA-II procedure, crowding distance metric and crowded tournament selection are described as sub-functions.

The 470 features described in section (3.5) are extracted from every amino acid sequence and PCA-NSGA-II is implemented for dimension reduction.

Algorithm 3 $PCA - NSGA - II$

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 $f_1, f_2 ... f_n$

Pareto fronts be denoted as $F_1, F_2, ... F_n$

$Gen = 0$

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 $F_1, F_2, \cdots, F_n$.

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 4 $NSGA-II(\ )$

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 $i \varepsilon P$.
for $(\forall j \neq i)$ *and* $j\ \varepsilon\ p$ **do**
  if $i \preceq j$ **then**
    Update $S_p = S_p \cup j$
  else $\{j \preceq i\}$
    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 $\forall\ i\ \varepsilon\ P_k$ *and* $\forall\ j\ \varepsilon\ S_i$ **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 5 *crowding distance( )*

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 $S_1, S_2 \cdots S_i$.

$|F_j| = l$ denotes number of solutions in a front.

cd_K denotes the crowding distance w.r.t K^{th} objective function.

$X_{[i,k]}$ represents i^{th} solution in the sorted list w.r.t K.

for $every\ fornt\ j = 1 \cdots R$ **do**

  for $every\ objective\ function\ M_1,\ M_2,\ \cdots, M_k$ **do**

    sort the solution in F_j in descending order.

    Assign $cd_K(x_{[1.k]}) = cd_K(x_{[i.k]}) = \infty$

    for $i = 2\ to\ l$ **do**

      assign $cd_K(x_{[i,k]}) = \frac{z_k(x_{[i+1,k]}) - z_k(x_{[i-1,k]})}{z_k^{max} - z_k^{min}}$

    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 6 *crowded tournament selection ( )*

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 $CD_i > CD_j$

end if

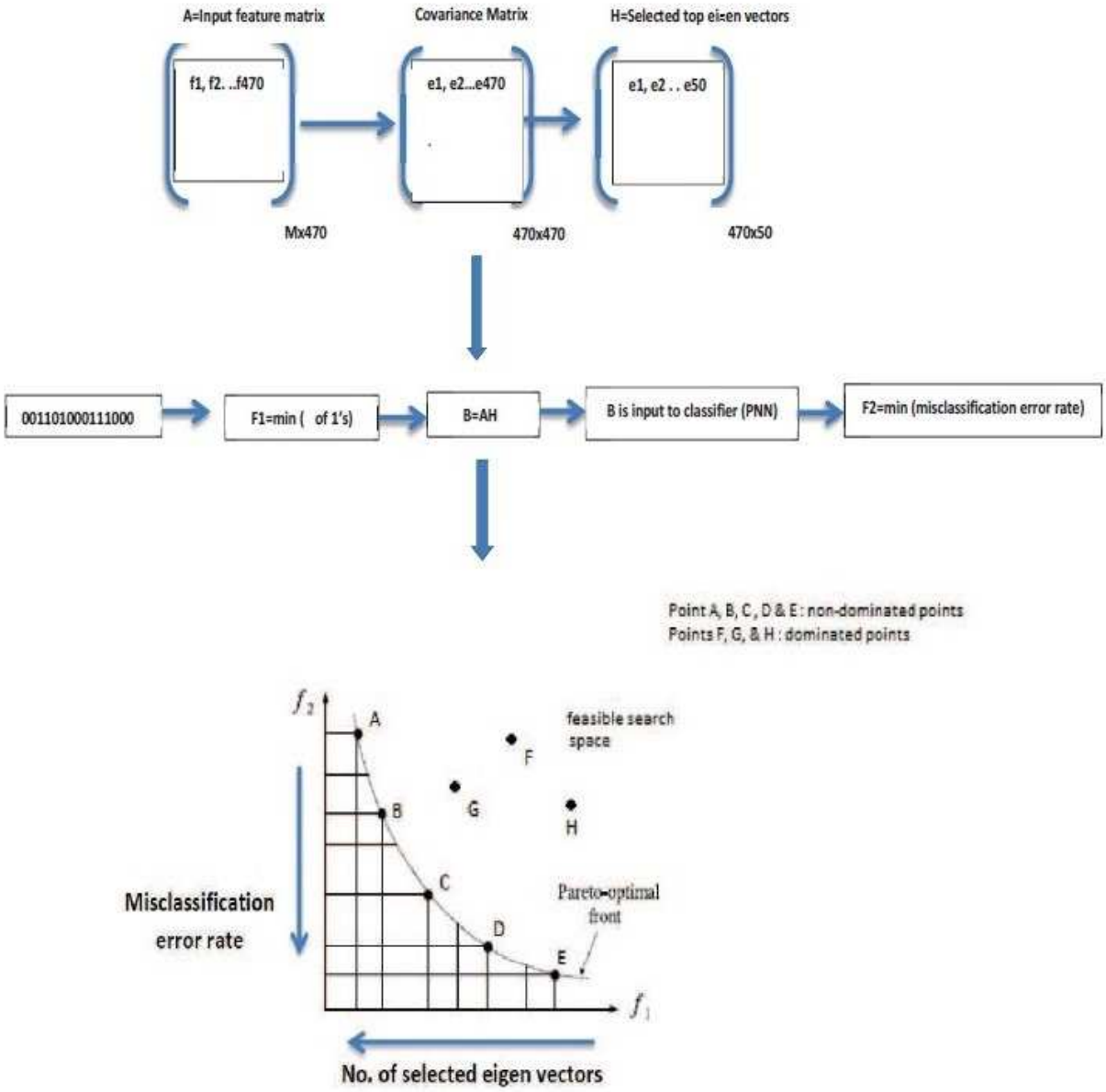

Figure 4.2: Brief overview of PCA-NSGA-II

4.4 Radial Basis Function Network as a Classifier

The basic concept and architecture of RBFN is described in section 3.2.

In generalized RBFN, the supervised learning of the center 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. R. Neruda and P. Kudova presented three learning methods of RBFN such as gradient based learning, three step unsupervised learning and evolutionary algorithms. The algorithms were examined on few benchmark problems. It was observed 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 [61].

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, S.Chen et al., used orthogonal least square (OLS) algorithm for the construction of RBFN and the forward regression procedure provided a systematic approach for supervised selection of centres in [62, 63]. Besides forward center selection, backward centre selection is proposed on batch OLS where the centres are sequentially removed while minimizing the network output error [64]. J.B.Gomm 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, non-linear time series and a real time multi-input and multi-output (MIMO) chemical process [65, 66].

4.4.1 Training of RBFN using ROLSA (Pseudocode)

The backward center selection algorithm proposed by J.B.Gomm et al. in [66] is implemented, to optimize the structure of RBFN. The pseudocode of ROLSA is described in Algorithm 7. 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.

Algorithm 7 *ROLSA*()

Let **N** = number of training samples

n_h= number of randomly chosen hidden centres.

Y=desired output matrix of size Nxp where p is the number of nodes in the output layer.

$\hat{Y}$= neural network output matrix of size N X p.

ϕ = hidden layer output matrix of size NXn_h

E= error matrix of size N X p.

W_{hp}= connecting weight between the hidden and output layer of size n_hXp.

Step1. Perform QR decomposition of ϕ matrix.

Step2. Evaluate $Q^TY = \begin{bmatrix} \hat{Y} \\ \tilde{Y} \end{bmatrix}$ where $\hat{Y}$ is of size n_h X p and $\tilde{y}$ is of size $(N - n_h)$ X p.

Step3. Evaluate the loss function $(V) = \left\|\tilde{Y}(N)\right\|_F^2 /(N)$.

Step4. Evaluate the Akaike's final prediction error $(FPEV) = \frac{1+\beta(n_h/N)}{1-\beta(n_h/N)}V$.

Step5. Remove each network center k and compute the loss functions.

$remove - hidden(\ )$

{

for *every hidden node* $i = 1$ *to* n_h **do**

$R(:, i) = [\ ]$

$[Q'R'] = qr(R)$

$OP = Q'\hat{Y}$

$\hat{Y}_j = op(1 : n_h - 1, :)$

$\tilde{y_j} = op(n_h, :)$

$ResK = \|\tilde{y_j}\|_F^2 + V/(N)$

$ResCK(1, i) = ResK$

end for

$[minval, minind] = min(ResCK)$

$FPEK = FPE * minval$

}

if $(FPEK < FPEV)$ **then**

$R(:, minind) = [\ ]$

$\hat{Y}(minind, :) = [\ ]$

$V = minval$

$FPEK = FPEV$

$n_h = n_h - 1$

$call\ remove - hidden(\ )$

else

Compute the final optimal weight matrix from the equation $R_j * W_j = \hat{Y}_j$

end if

4.5 Experiment Details

4.5.1 Input Details

The amino acid sequences are downloaded in FASTA format from UNIPROT repository (`http://www.uniprot.org/`). The four super-families considered for numerical simulations are Globin, Kinase, Ribitol dehydrogenase and Ligase.

4.5.2 Architecture of PNN as Inductive Algorithm

The basic architecture of PNN is shown in Figure 2.8. 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 build 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 to classify a test pattern. *50 randomly chosen samples* from every class formed the test matrix.

4.5.3 Parameter Details of PCA-NSGA-II

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 eigen vectors having non-zero eigen values. In this implementation, the chromosome string length was fixed at 63. Every one bit of chromosome represents selection of the eigen vector and zero represents discard of the eigen vector.

   The population size was fixed at N=40.
   P_c=0.7.

P_m=0.005.

2. Evaluation of fitness function: The first objective function
   (f_1): is to minimize the number of eigen vectors.
   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. This is calculated as the number of misclassified samples with respect to total number of samples.
   Therefore, $f_2 = (\frac{No.\ of\ misclassified\ samples}{200})$

3. Stopping criteria: The PCA-NSGA-II process terminates when 90% or more number of solutions are in the best non domination level.

4.5.4 Parameter Details of RBFN-ROLSA

From the number of solutions obtained from the *first pareto front (F1)*, *twenty-nine (29)*number of eigen vectors showing a misclassification error of ***0.57*** are selected. *Twenty five (25)* randomly chosen samples from every class of the training matrix were selected as hidden centres to form the the hidden layer of RBFN. The output layer consists of *four nodes*, each representing a class.

4.5.5 Parameters used for Measuring the Efficiency of Classifier:

The parameters used for measuring the efficiency of the classifier are

1. Precision=$[\frac{TP+TN}{TP+FP+TN+FN}] * 100\%$
2. Sensitivity $= \frac{TP}{TP+FN} * 100\%$
3. Specificity $= \frac{TN}{TN+FP} * 100\%$

The concept of TP, TN, FP, FN are described in section 2.4.

4.6 Results and Discussion

4.6.1 Results from First Stage

PCA-NSGA-II procedure took 2041 number of generations to meet the stopping criteria, thereby generating three pareto fronts (F1, F2, F3) shown in graph (Figure 4.3). Few solutions from the upper, lower and mid region of F1 are shown in the Table 4.1.

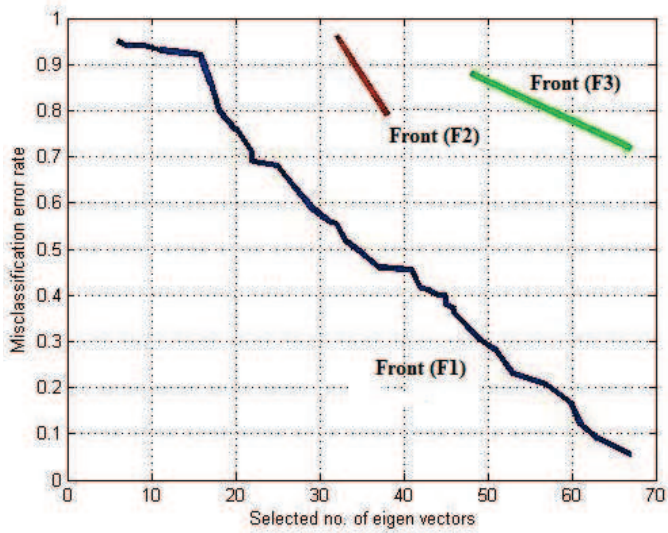

Figure 4.3: Pareto fronts generated after the Convergence of PCA-NSGA-II

Table 4.1: Pareto optimal solutions of first pareto front

Sl.No.	F1 (No. of eigen vectors)	F2 (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

4.6.2 Results from Second Stage

The 29 selected eigen vectors (from stage 1) showing a misclassification error rate of 0.57 are selected from the mid region of F1 which formed the reduced feature matrix. The implementation of Backward center selection algorithm (ROLSA) was implemented on $(29 \times 100 \times 4)$ RBF network. The removal of hidden centers continued till the termination criteria was met (as described in ROLSA pseudocode). Finally, the implementation of ROLSA yielded in $(29 \times 45 \times 4)$ reduced architecture. The final optimal weight matrix of size 45×4 on the reduced network architecture shows phenomenal performance in accuracy when applied on the test data set.

The number of correctly classified samples with respect to individual superfamily is shown in Table 4.2.

Table 4.2: No. of correctly classified samples from individual superfamily

Superfamily	Sensitivity in %	Specificity in %
Globin	98.21%	98.77%
Kinase	98.62%	98.61%
Rib.dehydro.	98.79%	98.54%
Ligase	98.89%	98.52%

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: $T_1, T_2, \cdots, T_{10}$. The ten RBFN classifiers are trained on the complement $\bar{T}_i$, and each classifier is then tested on the corresponding unseen test set T_i . The final cross-validation recognition rate (R) is given by :

$$R = \frac{1}{10} \sum_{i=1}^{10} r(T_i, \overline{T_i}) \tag{4.1}$$

where r(Ti; $\overline{T_i}$) is the recognition rate on T_i using the RBFN classifier trained on $\bar{T_i}$ [27, 67].

To show the efficiency of RBFN-ROLSA over standard neural networks, the comparison process is extended by implementing some standard neural networks. FFNN trained using BP algorithm, PNN and Standard RBFN (trained using supervised gradient descent learning algorithm) are implemented. The learning rate (η) and momentum (α) are the two main controlling parameters of BP algorithm. Keeping $\alpha = 0.5$ fixed and varying η in the range of 0.1 to 1, variation in performance accuracies were 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 neural networks are shown in Table 4.3.

Table 4.3: 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-AMOGA	97.91
6	RBFN-ROLSA	98.61

4.7 Conclusion

From the above implementation of two stage approach, it can be concluded that, PCA-NSGA-II gave the 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%** obtained from 10-fold cross-validation technique. This approach of structure optimization using ROLSA is very reliable in comparison to randomized evolutionary techniques. ROLSA has successfully trained RBFN to select optimal number of hidden centres as well as update the output layer weighting matrix. The RBFN so obtained is highly robust which works efficiently on large training and test data set. This approach can be applied to large data set with much lower requirements of computer memory. Thus, very small architecture having few number of hidden centres are obtained showing higher level of performance accuracy. 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 and also derived the most parsimonious structure of RBFN.

In the next chapter, the proposed method of using Multiobjective Genetic Algorithm and Support Vector Machine for Protein Superfamily Classification is discussed.

Chapter 5

Protein Superfamily Classification using Multiobjective Genetic Algorithm and Support Vector Machine

---

In this chapter, Multiobjective genetic algorithm (MOGA) and Support vector machine (SVM) are implemented for protein superfamily classification problem. MOGA using non-dominated sorting NSGA-II is used to select the optimal number of significant eigen vectors from the eigen space as well as optimize the hyper-parameters of SVM. In this GA based wrapper approach, the eigen vectors and the hyper-parameters of SVM are encoded in the chromosome. SVM classifier is wrapped with every chromosome for evaluating the fitness values. MOGA finds a solution to solve the trade-off problem between two conflicting objectives of SVM such as model complexity and accuracy of the classifier. To improve the convergence rate, AMOGA-SVM is imple-

mented and a comparative study between MOGA-SVM and AMOGA-SVM is performed.

5.1 Introduction

In previous chapters, the structure of RBFN is optimized using AMOGA and ROLSA respectively. But, generally any neural network suffers from major drawbacks [68], such as :

1. Greater computational burden.
2. Neural networks often converge on local minima rather than global minima.
3. Neural networks are prone to often over-fitting, which means, if training on a pattern goes on too long, then it may consider noise as part of pattern.

SVM doesn't suffer from either of these two drawbacks and have the following advantages over NN [69].

1. SVM have a regularization parameter as well as it is characterized by the number of support vectors rather than the dimensionality of the transformed space. So, SVM's tend to be less prone to the problem of over fitting.
2. SVMs provide a good out-of-sample generalization, if the parameters C and γ (in the case of a Gaussian kernel) are appropriately chosen. This means that, by choosing an appropriate generalization grade, SVMs can be robust, even when the training sample has some bias.
3. SVMs deliver a unique solution, since the optimality problem is convex. This is an advantage compared to neural networks, which give rise to

multiple solutions associated with local minima and for this reason NNs are not robust over different samples.

Thus, due to these advantages, SVM can be successfully implemented to perform classification task.

5.2 Related Work on Multiobjective Analysis of SVM

An introduction to SVM and multiple model estimation for non-linear classification are well described in [70,71]. The adaptation of kernel and regularization parameters of SVM by means of evolutionary optimization techniques are implemented in [72]. The SVM designed in this literature is evaluated on a real world pattern recognition task such as real time pedestrians detection in infra-red images for driver assistance systems. The two main objectives such as minimization of error and minimization of number of support vectors are casted as multi-objective problem for SVM model selection using RBF and Sigmoid kernels [73]. The SVM parameters and features are simultaneously optimized using genetic algorithm [74]. Real valued GA for optimizing the hyper-parameters of SVM was efficiently implemented for bankruptcy prediction, which was tested for the prediction of financial crisis [75]. The kernel parameter, input selection, ϵ-tube optimal dimension were used as decision variables of GA for SVM model construction which was then implemented for level predictions at variable time horizons for groundwater modelling in [76]. The concept of Genetic Programming (GP) was used to evolve a kernel for SVM classifier. The results were compared with standard SVM classifier using Polynomial, RBF and Sigmoid kernel with various parameter settings [77]. The combination of genetic algorithms (GAs) and all paired support vector machines (SVMs) for multi-class cancer identification was implemented in [78]. A robust gene selection approach based on a hybrid between GA and SVM is

shown in [79]. GA wrapped with SVM, derived feature reduction, which improved the hot method prediction accuracy. The technique of hot method prediction based optimization was potentially used in selective optimization [80]. Feature selection from amino acid sequence based on low relative entropy values using SVM as classifier was implemented in [23].

5.3 Brief Overview of the Entire Process

The SVM model selection problem is mapped as a multiobjective optimization problem, where the recognition rate and selection of number of eigen vectors from amino acid sequence are defined as two main objectives.

MOGA-SVM is a hybridized approach which tries to solve the trade-off problem by selecting optimal number of significant eigen vectors thereby maintaining a good level of performance accuracy of SVM. The encoding of chromosome becomes a very critical task as the size of feature vector extracted from amino acid sequence is too high. So to overcome this problem, eigen vectors having non-zero eigen values are encoded in the chromosome. Besides that, the regularization parameter C and the kernel parameters together constituting the hyper-parameters of SVM are also encoded in the chromosome. MOGA-SVM searches the eigen space to select the distinguishing eigen vectors. Based on the selected eigen vectors, the transformation matrix is built to map the original feature space to lower dimension feature space. Thus, the SVM model is constructed taking the input as the features from training data, and the hyperparameter values. The fitness of every chromosome is evaluated based on the mean recognition rate obtained from using 9 fold cross-validation technique and number of selected eigen vectors. Wrapper based approach is implemented to evaluate the quality of the selected eigenvectors by performing 9-fold cross-validation using the SVM model. After the construction of the model, the generalization error is calculated over the test data set. The optimal solutions are obtained from the pareto optimal set of the best non-dominated

level. The overview of the entire process is shown in Figure 5.1.

5.4 Non-Linear SVM

The basic concept of SVM was developed from Statistical Learning Theory and Structural Risk Minimization by Vapnik and Chervonenkis since the 1960s [81,82]. SVM has shown good performance in many applications such as bioinformatics, pattern recognition, image classification, cancer prediction, etc. The basic purpose to develop a SVM model is, to create a classifier by forming a linear separating hyperplane which maximizes the distance between two classes. For non-linearly separable data, SVM adopts two basic methods. First, the data are mapped into a rich feature space of high dimension using kernel function on non-linearly distributed data. Secondly, a soft margin hyperplane is introduced which adds a penalty function for violation of constraints to the optimization criterion. In other words, a hyperplane is constructed in the high dimensional space so that all other equations of hard margin remains the same. Thus, it is possible to find a linear optimal separating hyperplane in the new feature space using kernel function ϕ. The mapping of non-linearly separable data to higher dimension using kernel function is shown in Figure 5.2.

Suppose, the data is mapped to some other (possibly infinite dimensional) Euclidean space H, using a mapping function call *phi*. Then of course the training algorithm would only depend on the data through dot products in H, i.e. on functions of the form $\phi(x_i)\cdot\phi(x_j)$. Now if there were a "kernel function" K such that $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$, then it is only needed to use K in the training algorithm, and it is never needed to explicitly even know what ϕ is. The various kernel functions used for mapping are shown in Table 5.1.

First, the data is preprocessed by mapping the original non-linear input space to high dimension feature space using kernel function ϕ $(x \rightarrow \phi(x))$. After learning the mapping, the output becomes y: $f(x) = w.\phi(x) + b$. The width of the soft margin can be controlled by the penalty parameter C that

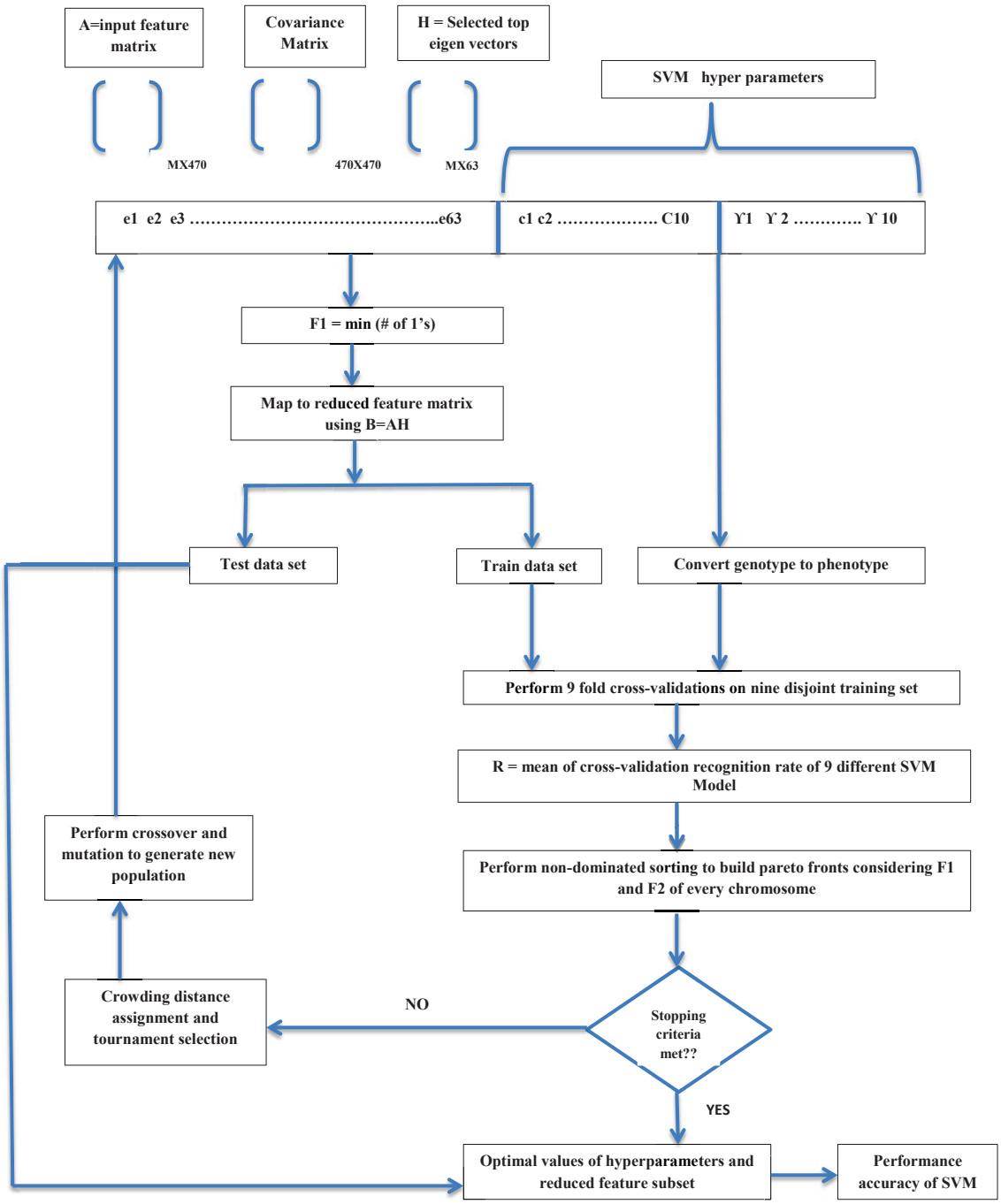

Figure 5.1: Brief overview of the entire process

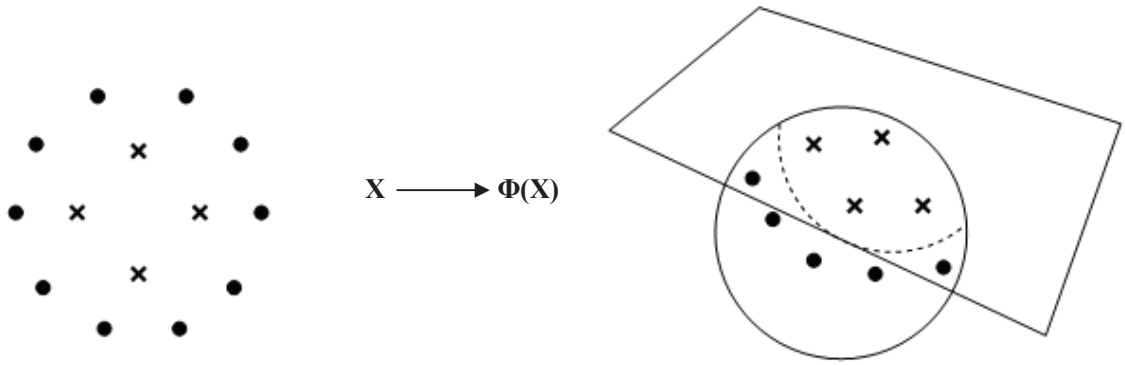

Figure 5.2: Mapping of non-linearly separable data to higher dimension using kernel function

Table 5.1: Various kernel functions used for mapping the non-linearly separable data to high dimension

Kernel function	description
Linear dot product kernel	$K(x, x_i) = (x^T.x_i)$
Polynomial kernel of degree d	$K(x, x_i) = [(x^T x_i) + 1]^d$
Gaussian RBF Kernel	$K(x, x_i) = exp(-\gamma \left\|x - x'\right\|^2)$
Hyperbolic Tangent (Sigmoid Kernel)	$K(x, x_i) = tanh(\alpha(x.x_i) + C)$
Inverse multiquadric function	$K(x, x_i) = \frac{1}{\sqrt{\left\|x-x_i\right\|^2+\beta}}$

determines the trade-off between the training error and VC dimension (Vapnik Chervonenkis dimension) of the model. Therefore, the equation becomes:

$$min \;\; \frac{1}{2}w^T w + C(no. \;\; of \;\; misclassified \;\; data) \tag{5.1}$$

where C is the penalty parameter, trading off the margin size (defined by $\|w\|$ i.e by $w^T w$) for the number of misclassified data points. The possible solution is to measure the distances ξ_i of the points crossing the margin and trade their sum for the margin size as given in Eqn. 5.2.

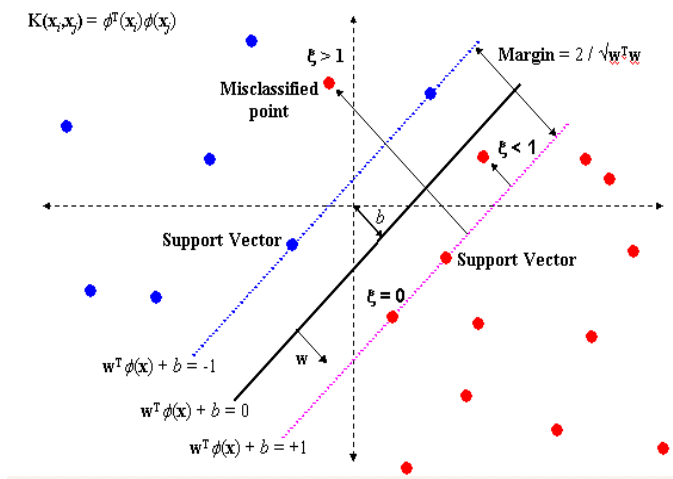

Figure 5.3: Representation of margin and support vectors in SVM

$$min \;\; \frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi_i \tag{5.2}$$

subject to constraints :

$$y_i[w^T x_i + b] \geq 1 - \xi_i, \;\; i = 1, \cdots l, \;\; \xi \geq 0 \tag{5.3}$$

i.e subject to :

$$[w^T x_i + b] \geq +1 - \xi_i, \quad for \;\; y_i = +1, \;\; \xi_i \geq 0 \tag{5.4}$$

$$[w^T x_i + b] \leq -1 + \xi_i, \quad for \;\; y_i = -1, \;\; \xi_i \geq 0 \tag{5.5}$$

The optimal solution to the above equations can be obtained by maximizing the variable α in the dual Lagrangian $L_d(\alpha)$ in the equation given below:

$$L_d(\alpha) = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j X_i^T X_j \tag{5.6}$$

To find the optimal hyperplane, the dual $L_d(\alpha)$ has to be maximized with respect to non-negative α_i and it should be smaller than or equal to C.

$$C \geq \alpha_i \geq 0, \;\; i = 1, \cdots, l \tag{5.7}$$

and under the constraints

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{5.8}$$

The penalty parameter C, is now the upper bound on α_i, which is determined by the user. The constant C is the regularization parameter which determines the trade-off between the margin and sum of slack variables ξ_i ($i = 1 \cdots l$).

In case of non-linearly separable data, the inner products in the dual Lagrangian $L_d(\alpha)$ is replaced by the Kernel functions. Therefore the non-linear objective function becomes:

$$Maximize \;\; L_d(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j (K(X_i, X_j)) \tag{5.9}$$

subject to $C \geq \alpha_i \geq 0, \; i = 1, \cdots, l$ and $\sum_{i=1}^{l} \alpha_i y_i = 0$.

The decision hyper-surface d(x) and the indicator function i_F for the non-linear SVM classifier is given by Eqn. 5.10 and 5.11.

$$d(x) = \sum_{i=1}^{l} y_i \alpha_i K(X_i, X_j) + b \tag{5.10}$$

$$i_F = sign(d(x)) = sign(\sum_{i=1}^{l} y_i \alpha_i K(X_i, X_j) + b) \tag{5.11}$$

The bias term b may be implicitly a part of the kernel function but for Gaussian RBF kernel b is not required [75].

The regularization parameter C controls the trade-off between model complexity and accuracy. The grid search method may be adopted to find the optimal hyper-parameter values by varying with a fixed step size. But the major drawback of this approach is, the computational complexity increases when the number of parameters to be optimized are high. Multi-objective evolutionary approach offers suitable solution for finding trade-off between several objectives. The best SVM model can be obtained from the pareto optimal

set obtained from the implementation of MOGA. Optimal values of hyper-parameter as well as optimal number of best distinguishing input features to the SVM model can be obtained by efficient exploration of the multi-modal search space.

5.5 SVM Kernel Parameter Selection and Feature Subset Selection using MOGA

The basic concept of MOGA is described in section 3.7. The schematic representation of pareto-optimality and dominated points is shown in Figure 3.2.

5.5.1 SVM Kernel Parameter Selection

SVM model selection involves the tuning of hyper-parameters which can be efficiently done by exploring the parameter search space. Here, the number of input features and the hyper-parameter values need to be optimized so as to reduce the generalization error of SVM model. To map the non-linearly separable data to high dimension, Gaussian Radial Basis Function (RBF) kernel is used, which is defined as:

$$K(x, x_i) = exp(-\gamma \left\| x - x_i \right\|^2) \tag{5.12}$$

where γ is the variance of the Gaussian RBF kernel. The regularization parameter C, determines the trade-off between the minimization of the fitting error and the minimization of the model complexity. To develop an efficient model C and γ need to be carefully determined.

5.5.2 Chromosome Design

Every chromosome has three parts. The first part represents the eigen vectors in eigen space, the second part encodes the regularization parameter C and the

third part encodes the variance of the Gaussian RBF Kernel γ. The schematic representation of chromosome is shown in Figure 5.3. The entire chromosome is binary encoded which consists of strings of 0's and 1's. The length of the first part depends on the total number of eigen vectors having non-zero eigen values. Binary bit 1 represents the selection of eigen vector and 0 represents the discard of eigen vector. The presence or absence of eigen vector changes as generation evolves. The number of bits in the second and third part i.e $n_c,\ and\ n_\gamma$ are calculated according to number of precision required which is given in Eqn. 5.13.

$$(\beta_i - \alpha_i) * 10^\gamma + 1 \leq 2^{n_i} \tag{5.13}$$

where β_i = maximum range of the parameter.
α_i= minimum range of the parameter.
γ= number of precision required after decimal point.
n_i = number of bits in the chromosome string.

For the second and third part, the genotype to phenotype conversion is obtained by the Eqn. 5.14.

$$p = min_p + \frac{max_p - min_p}{2^l - 1} * d \tag{5.14}$$

where p= phenotype of the string.
min_p = minimum value of the parameter.
max_p = maximum value of the parameter.
d= decimal value of bit string.
l=length of the string.

5.5.3 Fitness Function

The evaluation function used here consists of two fitness functions, f_1 and f_2. The first fitness function f_1 is based on the minimization of number of eigenvectors selected. The second fitness function f_2 is evaluated to minimize

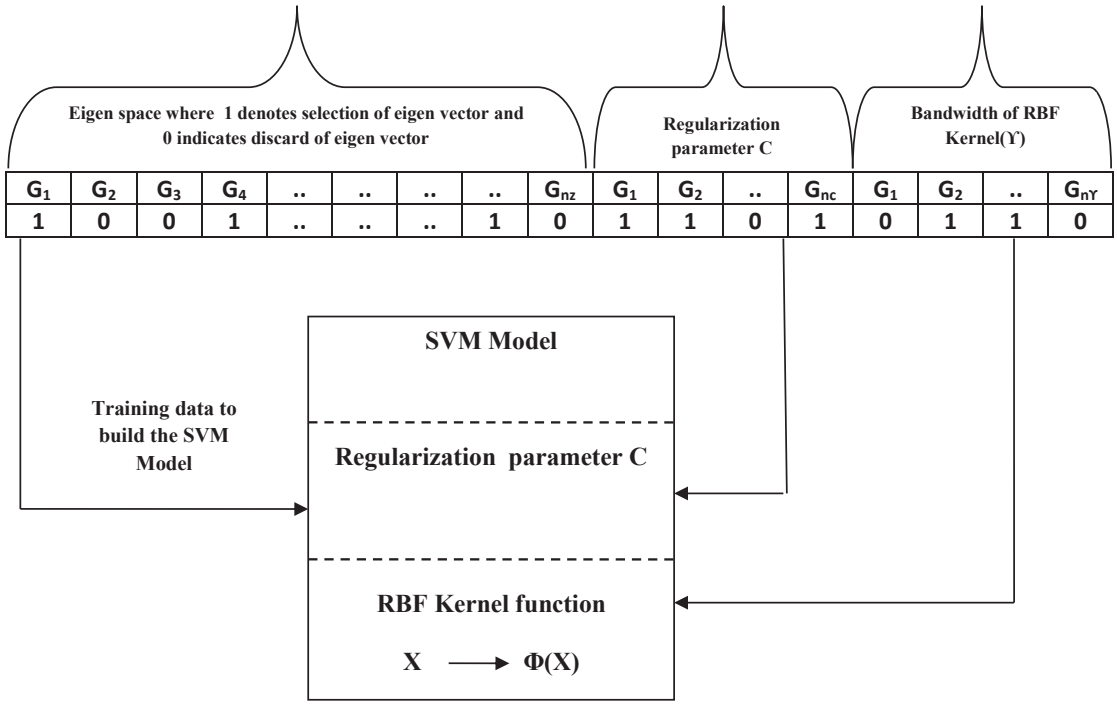

Figure 5.4: Representation of chromosome

(1-R) where R denotes the mean recognition rate of the SVM models obtained from 9-fold cross validation technique. The training set T is randomly partitioned into nine equal disjoint sets: $T_1, T_2, \cdots, T_9$. The nine SVM classifiers are trained on the complement T_i, and each classifier is then tested on the corresponding unseen test set T_i [27]. The final cross-validation recognition rate is given by:

$$R = \frac{1}{9}\sum_{i=1}^{9} r(T_i, \bar{T}_i) \tag{5.15}$$

Based on two fitness values obtained for every chromosome, the non-dominated sorting is performed to obtain pareto fronts at various levels.

5.6 MOGA-SVM

MOGA-SVM aims to optimize the feature subset as well as the hyper-parameters of SVM. The population size is fixed at N for every generation. The probabilities of crossover and mutation are the two main controlling parameters of MOGA. The evaluation of two fitness functions and the generation of pareto fronts using non-dominated sorting is shown in Algorithm 9. The calculation of crowding distance and tournament selection are shown as sub-functions.

Algorithm 8 $MOGA - SVM$

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 $f_1, f_2 ... f_n$

Pareto fronts be denoted as $F_1, F_2, ... F_m$

$Data_{set} = \{D_{train}\ and\ D_{test}\}$

$Gen = 0$

repeat

Step 1. $Gen = Gen + 1$.

Step 2. Initialize population P_0.

Initialize N number of chromosomes as random individuals which are encoded as binary strings. The chromosome is divided into three parts and the length of the first part depends on the total number of 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}. The second part encodes regularization parameter and the third part encodes variance of Gaussian RBF kernel.

Step 3. Evaluate fitness function f_1= minimize number of 1's in the chromosome string in the first part.

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. For the second and third part of chromosome, convert genotype to phenotype using the equation : $p = min_p + \frac{max_p - min_p}{2^l - 1} * d$

Step 6. Use the 9-fold cross-validation technique to build 9 SVM models considering training set, $T_i, i = 1, \cdots 9$. Evaluate mean recognition rate as, $R = \frac{1}{9}\sum_{i=1}^{9} r(T_i, \bar{T}_i)$.

Step 7. Evaluate second fitness function as $f_2 = minimize\ (1 - R)$.

Step 8. Considering f_1 and f_2, perform non-dominated sorting using NSGA-II( ) and generate pareto fronts such as $F_1, F_2, \ \cdots\ , F_m$.

Step 9. Calculate the crowding distance of all solution points using the crowding distance( ).

Step 10. Perform tournament selection by selecting N random pairs from P_t.

Step 11. Use the crowded comparison operator( ) to select the most widely spread solutions.

Step 12. Perform pairwise crossover and bitwise mutation to create new offspring.

Step 13. Let the new population be denoted as P_{t+1}.

until $(|F_1| \geq 90\%\ of\ N)$

Algorithm 9 $NSGA-II(\ )$

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 $i \varepsilon P$.
for $(\forall j \neq i)$ *and* $j\ \varepsilon\ p$ **do**
  if $i \preceq j$ **then**
    Update $S_p = S_p \cup j$
  else $\{j \preceq i\}$
    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 $\forall\ i\ \varepsilon\ P_k$ *and* $\forall\ j\ \varepsilon\ S_i$ **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 10 *crowding distance( )*

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 $S_1, S_2 \cdots S_i$.

$|F_j| = l$ denotes number of solutions in a front.

cd_K denotes the crowding distance w.r.t K^{th} objective function.

$X_{[i,k]}$ represents i^{th} solution in the sorted list w.r.t K.

for *every* $fornt\ j = 1 \cdots R$ **do**

  for *every objective* $function\ M_1,\ M_2,\ \cdots, M_k$ **do**

    sort the solution in F_j in descending order.

    Assign $cd_K(x_{[1.k]}) = cd_K(x_{[i.k]}) = \infty$

    for $i = 2$ *to* l **do**

      assign $cd_K(x_{[i,k]}) = \frac{z_k(x_{[i+1,k]}) - z_k(x_{[i-1,k]})}{z_k^{max} - z_k^{min}}$

    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 11 *crowded tournament selection ( )*

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 $CD_i > CD_j$

end if

5.7 Experiment Details

5.7.1 Input Details

The amino acid sequences are downloaded in FASTA format from UNIPROT repository (`http://www.uniprot.org/`). The four super-families considered for numerical simulations are Globin, Kinase, Ribitol dehydrogenase and Ligase.

5.7.2 Parameter Details of PCA-SVM-NSGA-II

Listed below are the descriptions for the parameters used in PCA-SVM-NSGA-II.

1. Initialization of chromosome:

   The population is initialized by encoding the chromosomes as strings of 0's and 1's. The first part of chromosome length is fixed on the basis of number of eigen vectors having non-zero eigen values. In this experiment, the chromosome string length is fixed at 63. Every one bit of chromosome represents selection of the eigen vector and zero represents discard of the eigen vector. The second part and third part have 10 bits each based on the order of precision.

   The population size was fixed at N=60.
   Probability of Crossover (P_c=0.7).
   Probability of Mutation (P_m=0.005).
   n_c *and* n_γ = 10 bits each.
   γ range is varied within 0 to 1 and order of precision is 3.
   C range is varied within 1 to 10 and order of precision is 2.

2. Evaluation of fitness function:

   The first fitness function:
   (f_1): is to minimize the number of eigen vectors.

f_1 is computed as number of 1's in first part of the chromosome string.

The second fitness function:

(f_2): is to minimize (1-R) where R denotes $R = (\frac{1}{9}\sum_{i=1}^{9} r(T_i, \bar{T}_i))$

3. Stopping criteria: AMOGA-SVM and MOGA-SVM are implemented to obtain the best pareto front. Both the processes are run for **1000 generations**.

5.8 Results and Discussion

AMOGA-SVM and MOGA-SVM were run for 1000 generations and to evaluate their performances graphs were plotted for both the algorithms. The graphs obtained from numerical simulations shows that the number of scattered pareto points gets slowly converge to the best non-domination level as evolution takes place. But, number of scattered pareto points in MOGA is more in comparison to AMOGA. The best pareto front obtained after 1000 generations gave rise to number of solutions which are stored as pareto optimal set. Every solution in the set resulted into various ensembles of SVM model which is formed based on input feature set, C and γ values. The optimal solution can be selected from the mid region of the pareto front, thereby solving the trade-off problem between model complexity and accuracy. Four best solutions from the mid region of pareto front obtained from MOGA and AMOGA were selected to construct four different ensembles of SVM model. The number of correctly classified samples from individual superfamily on the test data set are shown in Table 5.2 and Table 5.3 respectively.

Table 5.2: Pareto optimal subset from first pareto front of MOGA after 1000 generations

Sl. No.	f_1 **(No. of Eigen vectors)**	**C**	γ	$f_2 = mean \quad validation \quad error$	Test accuracy in (%)
1	30	4.64	0.483	0.0230	97.5
2	25	4.42	0.612	0.0275	97.1
3	22	3.65	0.518	0.0290	96.4
4	18	4.75	0.672	0.0340	95.8

Table 5.3: Pareto optimal subset from first pareto front of AMOGA after 1000 generations

Sl. No.	f_1 **(No. of Eigen vectors)**	**C**	γ	$f_2 = mean \quad validation \quad error$	Test accuracy in (%)
1	32	2.42	0.446	0.0120	98.8
2	25	3.85	0.582	0.0150	98.6
3	23	3.15	0.495	0.0180	97.4
4	17	2.75	0.510	0.0220	97.1

Table 5.4: 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-AMOGA	97.91
6	RBFN-ROLSA	98.61
7	MOGA-SVM	97.5
8	AMOGA-SVM	98.8

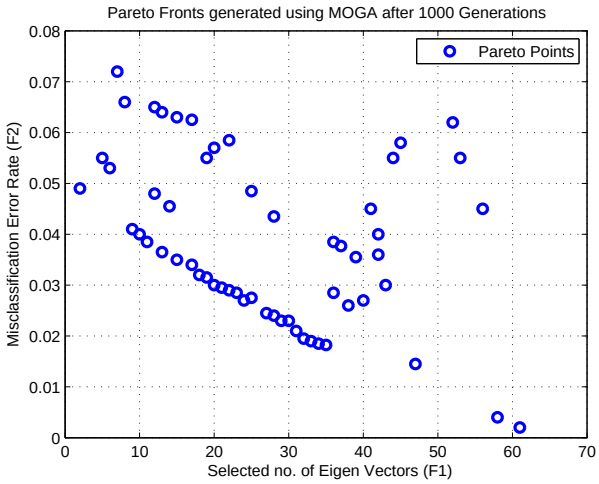

Figure 5.5: Performance of MOGA after 1000 Generations

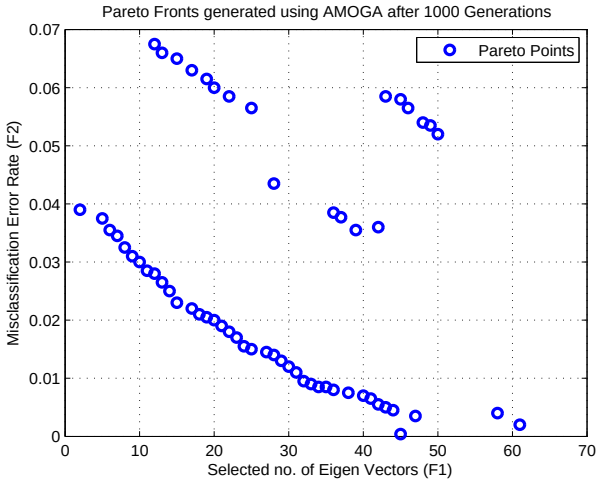

Figure 5.6: Performance of AMOGA after 1000 Generations

5.9 Conclusion

From the above implementation of MOGA-SVM and AMOGA-SVM, it can be concluded that the hyper-parameters of SVM and feature subset can be optimized simultaneously. The selection of optimal number of significant features affects the performance accuracy of the classifier. The MOGA based approach using Gaussian RBF kernel has shown promising results by selecting very few input features to the SVM model. Assuming the same stopping criteria for both algorithms, it is observed that AMOGA converged faster than MOGA. The maximum test accuracy reached upto **98.8%.** using AMOGA-SVM and **97.5%** using MOGS-SVM. Thus, the technique can be successfully applied for the problem of protein superfamily classification.

The next chapter concludes the thesis and summarizes all our proposed methods and also provides scope for further research.

Chapter 6

Conclusions

Protein superfamily classification problem is of great concern for drug analyst. If a newly discovered protein responsible for the cause of new disease gets correctly classified to its superfamily, then the task of the drug analyst becomes simpler. The analysts can recombine some existing drugs to discover new drugs or else he may search the ligand database for finding out the ligand-protein pair for that particular superfamily. Thus, the search process is enormously reduced, as database is searched for one superfamily. So, correct classification of protein, greatly matters for the discovery of appropriate drugs. The application of computational intelligent techniques offers promising solutions to handle and manipulate the long dimensional protein data as they are robust and possess the ability of tolerance for imprecision, uncertainty, approximate reasoning, and partial truth.

To start with, in this thesis, an elaborate survey on the literature available for the problem of protein superfamily classification has been done, under three main phases. In the first phase, the various global and local features extracted from amino acid sequence are reviewed and the dimension reduction techniques applied to reduce the long dimension vector are studied. SVD and PCA algorithms are implemented and the efficiency is measured using

PNN as classifier. In the second phase, an elaborate study on various classifiers implemented for the classification task was done. FNN trained using BP algorithm was implemented as a classifier but the major drawback of BP algorithm is that, it takes long and uncertain training time to converge and may get stuck in local minima. To overcome the drawback, evolutionary optimization techniques were applied to fasten the training process of FFNN. GA, AGA, PSO, MOPSO, and DE were implemented and it was observed that, AGA-BP outperformed the other evolutionary techniques. Besides that, three standard neural networks such as FFNN, PNN and RBFN were implemented. Although PNN performed well; but, the major drawback of PNN is that, it performs well when the training data set is small in size. But as the size of training data set increases, the architecture of PNN becomes very large and complex, as every node in the pattern layer represents a training sample. So, RBFN was the best choice among standard neural networks to perform the classification task.

In case of RBFN, the problem of finding the number of hidden centers is a very critical issue in the design of RBFN. The number of basis function controls the complexity and generalization ability of the network. AMOGA, a variation of MOGA, was implemented to derive the optimal architecture of RBFN. The modification to the earlier approach of MOGA was done based on the two key controlling parameters such as probability of crossover and probability of mutation. These values were adaptively varied based on the performance of the algorithm i.e., based on the percentage of total population present in the best non-domination level. The RBFN obtained from the pareto optimal set, has shown good classification accuracy in comparison to standard neural networks. This was validated by performing classification considering four protein superfamilies. AMOGA outperformed MOGA in terms of speed and accuracy by protecting the high fitness solutions from getting disrupted in the search space. PCA was used for dimension reduction of long feature vectors.

The approach of selecting top few eigen vectors (as in traditional PCA) may not always be the right approach for feature extraction. The eigen vectors having low eigen values may also have a good impact on the performance accuracy of the classifier. So, the proposed algorithm PCA-NSGA-II searches the eigen space to select the distinguishing eigen vectors. The non-dominated solutions obtained from the pareto front solves the trade-off problem by compromising between the number of eigen vectors selected and the accuracy obtained by the classifier. For efficient design of classifier, although AMOGA performed well in terms of convergence speed and accuracy, but it gave rise to optimized structure of RBFN. To derive the most parsimonious structure of RBFN and improve the performance accuracy, RBFN-ROLSA was implemented. Thus, ROLSA derived very small architecture of RBFN having few number of hidden centres showing high level of performance accuracy.

Generally, any neural network suffers from two major drawbacks. They often converge in local minima rather than global minima. The problem of over-fitting is possible in neural networks, which means, if training on a pattern goes on for too long time, then it may consider noise as part of pattern. To overcome these drawbacks, SVM was preferred to be implemented, for classification. MOGA-SVM solved the trade-off problem between model complexity and accuracy of the SVM model. To further improve the performance of MOGA-SVM, AMOGA-SVM was implemented. It was observed from numerical simulations that, AMOGA converged faster towards the global pareto front. The AMOGA based approach using Gaussian RBF kernel has shown promising results by selecting very few input features to the SVM model. The method was cross-validated by performing numerical simulations considering four protein superfamilies on nine different disjoint data sets. Therefore, from above implementation it can be concluded that, the hyper-parameters of SVM and feature subset can be optimized simultaneously.

Scope for Further Research

The research findings made out of this thesis, has opened several auxiliary research directions, which can be further investigated. Further research can be carried out to develop concepts and techniques for optimal feature selection and extraction from amino acid sequences. Some other evolutionary optimization techniques can be applied for significant feature extraction. In this thesis, RBFN and SVM models are used as the classifier. Some more investigation on SVM model can be carried out to further improve its performance. The work of protein superfamily classification can further be extended for disease prediction and remote homology detection.

Annexure

Annexure - I

List of Amino Acids

Alanine	Ala	A
Arginine	Arg	R
Asparagine	Asn	N
Aspartic acid	Asp	D
Cysteine	Cys	C
Glutamic acid	Glu	E
Glutamine	Gln	Q
Glycine	Gly	G
Histidine	His	H
Isoleucine	Ile	I
Leucine	Leu	L
Lysine	Lys	K
Methionine	Met	M
Phenylalanine	Phe	F
Proline	Pro	P
Serine	Ser	S
Threonine	Thr	T
Tryptophan	Trp	W
Tyrosine	Tyr	Y
Valine	Val	V

Annexure II

Sample Input Feature Matrix (Protein Data)

	1	*2*	*3*	*4*	*5*	.	.	.	*96*	*97*	*98*	*99*	*100*	.	.	*464*	*465*	*466*	*467*	*468*	*469*	*470*
1	48871.4	7.078159	2170	3468	650	.	.	.	0	0	0	1	1	.	.	4	3	5	2	9	6	2
2	48871.4	7.078159	2170	3468	650	.	.	.	0	0	0	1	1	.	.	4	3	5	2	9	6	2
3	117780.5	5.033166	5249	8184	1410	.	.	.	14	0	6	8	3	.	.	13	12	20	0	32	20	6
4	14383.15	9.074627	676	1042	162	.	.	.	0	0	0	0	0	.	.	6	2	3	2	4	4	3
5	48871.4	7.078159	2170	3468	650	.	.	.	0	0	0	1	1	.	.	4	3	5	2	9	6	2
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
55	58421.05	7.962265	2674	4110	690	.	.	.	3	0	2	3	1	.	.	20	9	15	0	14	13	13
56	42323.37	6.215256	1914	3037	505	.	.	.	2	0	2	4	2	.	.	9	5	10	0	10	6	4
57	59260.35	7.565567	2837	4197	653	.	.	.	1	0	1	0	0	.	.	30	7	8	0	36	21	18
58	10471.91	9.658272	467	696	136	.	.	.	0	0	0	0	0	.	.	1	3	3	0	3	2	2
59	16487.99	9.771599	739	1183	199	.	.	.	1	0	0	0	1	.	.	5	1	1	0	5	4	2
60	15789.16	8.909149	712	1145	191	.	.	.	1	0	0	1	0	.	.	3	1	0	0	4	5	2
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
396	16142.54	9.491379	718	1184	198	.	.	.	1	0	0	0	0	.	.	2	2	1	0	1	2	1
397	17758.22	8.447296	783	1262	228	.	.	.	0	0	0	0	0	.	.	6	1	2	0	5	1	1
398	15903.16	4.737358	710	1135	189	.	.	.	1	0	1	0	0	.	.	4	2	2	0	4	2	1
399	16884.52	10.56083	753	1219	215	.	.	.	1	0	0	0	0	.		4	2	1	0	3	5	0
400	16955.43	10.06843	752	1204	214	.	.	.	1	0	0	0	0	.	.	4	2	1	0	4	4	1

Bibliography

[1] S. Mitra and Y. Hayashi, "Bioinformatics with soft computing," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 36, no. 5, pp. 616–635, 2006.

[2] H. Bhaskar, D. C. Hoyle, and S. Singh, "Machine learning in bioinformatics: A brief survey and recommendations for practitioners," *Computers in biology and medicine*, vol. 36, no. 10, pp. 1104–1125, 2006.

[3] A. M. Campbell and L. J. Heyer, *Discovering genomics, proteomics, and bioinformatics*. Benjamin Cummings San Francisco:, 2003.

[4] J. Sourirajan, "Protein structure prediction," Tech. Rep., 2004.

[5] D. B. Kitchen, H. Decornez, J. R. Furr, and J. Bajorath, "Docking and scoring in virtual screening for drug discovery: methods and applications," *Nature reviews Drug discovery*, vol. 3, no. 11, pp. 935–949, 2004.

[6] B. Liu, X. Wang, L. Lin, Q. Dong, and X. Wang, "A discriminative method for protein remote homology detection and fold recognition combining top-n-grams and latent semantic analysis," *BMC bioinformatics*, vol. 9, no. 1, p. 510, 2008.

[7] A. Ben Hur and D. Brutlag, "Remote homology detection: a motif based approach," *Bioinformatics*, vol. 19, no. suppl 1, pp. 26–33, 2003.

[8] 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.

[9] W. L. Jorgensen, "Rusting of the lock and key model for protein-ligand binding," *Science*, vol. 254, no. 5034, pp. 954–955, 1991.

[10] L. Zheng and X. He, "Classification techniques in pattern recognition," Tech. Rep., 2005.

[11] T. M. Mitchell, *Machine learning*. McGraw Hill, 1997.

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

[13] R. Jensen and Q. Shen, *Computational intelligence and feature selection: rough and fuzzy approaches*. Wiley-IEEE Press, 2008, vol. 8.

[14] M. Dash and H. Liu, "Feature selection for classification," *Intelligent data analysis*, vol. 1, no. 1-4, pp. 131–156, 1997.

[15] M. Ben-Bassat, "Pattern recognition and reduction of dimensionality," *Handbook of Statistics*, vol. 2, pp. 773–910, 1982.

[16] S. Sharma, V. Kumar, T. Sobha Rani, S. Durga Bhavani, and S. Bapi Raju, "Application of neural networks for protein sequence classification," in *Intelligent Sensing and Information Processing, 2004. Proceedings of International Conference on*. IEEE, 2004, pp. 325–328.

[17] 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.

[18] 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.

[19] 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.

[20] C. H. Wu and T.-C. Chang, "Protein classification using a neural network database system," in *Proceedings of the conference on Analysis of neural network applications*. ACM, 1991, pp. 29–41.

[21] S. Mohamed, D. Rubin, and T. Marwala, "Multi-class protein sequence classification using fuzzy artmap," in *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, vol. 2. IEEE, 2006, pp. 1676–1681.

[22] C. Chothia and A. V. Finkelstein, "The classification and origins of protein folding patterns," *Annual review of biochemistry*, vol. 59, no. 1, pp. 1007–1035, 1990.

[23] J.-X. D. Xing-Ming Zhao1 and H. qiang Wang, "A new technique for extracting features from protein sequences," in *proceedings of International Conference on Intelligent Computing*, 2005, pp. 1223–1232.

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

[25] 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.

[26] 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.

[27] X.-M. Zhao, Y.-M. Cheung, D.-S. Huang *et al.*, "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.

[28] M. L. Raymer, W. F. Punch, E. D. Goodman, L. A. Kuhn, and A. K. Jain, "Dimensionality reduction using genetic algorithms," *Evolutionary Computation, IEEE Transactions on*, vol. 4, no. 2, pp. 164–171, 2000.

[29] P. N. Rao, T. U. Devi, D. Kaladhar, G. Sridhar, and A. A. Rao, "A probabilistic neural network approach for protein superfamily classification," *Journal of Theoretical and Applied Information Technology*, vol. 6, no. 1, pp. 101–105, 2009.

[30] E. A. FerrAn, P. Ferrara, and B. Pflugfelder, "Protein classification using neural networks," in *Proc. First International Conference on Intelligent Systems for Molecular Biology*, 1993, pp. 127–135.

[31] M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 24, no. 4, pp. 656–667, 1994.

[32] D. Wang, N. Lee, T. S. Dillon, and N. J. Hoogenraad, "Protein sequences classification using modular rbf neural networks," in *AI 2002: Advances in Artificial Intelligence*. Springer, 2002, pp. 477–486.

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

[34] D. F. Specht, "Probabilistic neural networks," *Neural networks*, vol. 3, no. 1, pp. 109–118, 1990.

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

[36] E. Mansouri, S. Katebi, H. Mohabatkar, R. Boustani, and A. M. Sadar, "Generating fuzzy rules for protein classification," *Iranian Journal of Fuzzy Systems*, vol. 5, no. 2, pp. 21–33, 2008.

[37] D. Wang, N. K. Lee, T. S. Dillon *et al.*, "Extraction and optimization of fuzzy protein sequences classification rules using grbf neural networks," *Neural Information Processing-Letters and Reviews*, vol. 1, no. 1, pp. 53–57, 2003.

[38] L. French, A. Ngom, and L. Rueda, *Fast protein superfamily classification using principal component null space analysis*. Springer, 2005.

[39] S. Haykin and N. Network, "A comprehensive foundation," *Neural Networks*, vol. 2nd edition, pp. 320–325, 2009.

[40] G. G. Yen and H. Lu, "Hierarchical rank density genetic algorithm for radial-basis function neural network design," *International Journal of Computational Intelligence and Applications*, vol. 3, no. 3, pp. 213–232, 2003.

[41] O. Buchtala, M. Klimek, and B. Sick, "Evolutionary optimization of radial basis function classifiers for data mining applications," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 35, no. 5, pp. 928–947, 2005.

[42] B. Yu and X. He, "Training radial basis function networks with differential evolution," in *IEEE International Conference on Granular Computing*. Citeseer, 2006, pp. 369–372.

[43] N. Kondo, T. Hatanaka, and K. Uosaki, "Nonlinear dynamic system identification based on multiobjectively selected rbf networks," in *Computational Intelligence in Multicriteria Decision Making, IEEE Symposium on*. IEEE, 2007, pp. 122–127.

[44] N. Kondo, K. Uosaki, and T. Hatanaka, "Rbf networks ensemble construction based on evolutionary multi-objective optimization," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 12, no. 3, pp. 297–303, 2008.

[45] S. Qasem and S. Shamsuddin, "Generalization improvement of radial basis function network based on multi-objective particle swarm optimization," *J. Artif. Intell*, vol. 3, no. 1, 2010.

[46] S. A. Billings and G. L. Zheng, "Radial basis function network configuration using genetic algorithms," *Neural Networks*, vol. 8, no. 6, pp. 877–890, 1995.

[47] 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.

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

[49] T. Hatanaka, N. Kondo, and K. Uosaki, "Multi-objective structure selection for radial basis function networks based on genetic algorithm," in

Evolutionary Computation, 2003. CEC'03. The 2003 Congress on, vol. 2. IEEE, 2003, pp. 1095–1100.

[50] 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.

[51] S. N. Qasem and S. M. Shamsuddin, "Radial basis function network based on time variant multi-objective particle swarm optimization for medical diseases diagnosis," *Applied Soft Computing*, vol. 11, no. 1, pp. 1427–1438, 2011.

[52] M. M. Zirkohi, M. M. Fateh, and A. Akbarzade, "Design of radial basis function network using adaptive particle swarm optimization and orthogonal least squares," *Journal of Software Engineering and Applications*, vol. 3, no. 7, pp. 704–708, 2010.

[53] A. P. Engelbrecht, *Computational intelligence: an introduction.* Wiley Press, 2007.

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

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

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

[57] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Reliability Engineering & System Safety*, vol. 91, no. 9, pp. 992–1007, 2006.

[58] K. Balci and V. Atalay, "Pca for gender estimation: which eigenvectors contribute?" in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 3. IEEE, 2002, pp. 363–366.

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

[60] P. Langley, "Selection of relevant features in machine learning," Tech. Rep., 1994.

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

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

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

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

[65] D. Yu, J. 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.

[66] J. B. Gomm and D. L. Yu, "Selecting radial basis function network centers with recursive orthogonal least squares training," *IEEE Trans. Neur. Netw.*, vol. 11, no. 2, pp. 306–314, 2000.

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

[68] J. V. Tu, "Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes," *Journal of clinical epidemiology*, vol. 49, no. 11, pp. 1225–1231, 1996.

[69] L. Auria and R. A. Moro, "Support vector machines (svm) as a technique for solvency analysis," Tech. Rep., 2008.

[70] V. Kecman, "Support vector machines–an introduction," in *Support vector machines: theory and applications*. Springer, 2005, pp. 1–47.

[71] Y. Ma and V. Cherkassky, "Multiple model estimation for nonlinear classification," in *Support Vector Machines: Theory and Applications*. Springer, 2005, pp. 49–76.

[72] T. Suttorp and C. Igel, "Multi-objective optimization of support vector machines," in *Multi-objective machine learning*. Springer, 2006, pp. 199–220.

[73] G. Narzisi, "An experimental multi-objective study of the svm model selection problem," Tech. Rep.

[74] C.-L. Huang and C.-J. Wang, "A ga-based feature selection and parameters optimizationfor support vector machines," *Expert Systems with applications*, vol. 31, no. 2, pp. 231–240, 2006.

[75] C.-H. Wu, G.-H. Tzeng, Y.-J. Goo, and W.-C. Fang, "A real-valued genetic algorithm to optimize the parameters of support vector machine for predicting bankruptcy," *Expert Systems with Applications*, vol. 32, no. 2, pp. 397–408, 2007.

[76] O. Giustolisi, "Using a multi-objective genetic algorithm for svm construction." *Journal of Hydroinformatics*, vol. 8, pp. 125–139, 2006.

[77] T. Howley and M. G. Madden, "The genetic kernel support vector machine: Description and evaluation," *Artificial Intelligence Review*, vol. 24, no. 3-4, pp. 379–395, 2005.

[78] S. Peng, Q. Xu, X. B. Ling, X. Peng, W. Du, and L. Chen, "Molecular classification of cancer types from microarray data using the combination of genetic algorithms and support vector machines," *FEBS letters*, vol. 555, pp. 358–362, 2003.

[79] L. Li, W. Jiang, X. Li, K. L. Moser, Z. Guo, L. Du, Q. Wang, E. J. Topol, Q. Wang, and S. Rao, "A robust hybrid between genetic algorithm and support vector machine for extracting an optimal feature gene subset," *Genomics*, vol. 85, no. 1, pp. 16–23, 2005.

[80] S. Johnson and V. Shanmugam, "Feature subset selection for hot method prediction using genetic algorithm wrapped with support vector machines," *Journal of Computer Science*, vol. 7, no. 5, pp. 707–714, 2011.

[81] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[82] V. Vapnik, *The nature of statistical learning theory.* Springer, 1999.

[83] A.-E. Hassanien, M. G. Milanova, T. G. Smolinski, and A. Abraham, *Computational intelligence in solving bioinformatics problems: Reviews, perspectives, and challenges.* Springer, 2008.

[84] R. K. Jena, M. M. Aqel, P. Srivastava, and P. K. Mahanti, "Soft computing methodologies in bioinformatics," *European Journal of Scientific Research*, vol. 26, no. 2, pp. 189–203, 2009.

[85] S. Vipsita, B. K. Shee, and S. K. Rath, "An efficient technique for protein classification using feature extraction by artificial neural networks," in *India Conference (INDICON), 2010 Annual IEEE*. IEEE, 2010, pp. 1–5.

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

Dissemination

Journal

1. S. Vipsita and S. K. Rath, "Protein superfamily classification using adaptive evolutionary radial basis function network," *International Journal of Computational Intelligence and Applications*, vol. 11, no. 4, World Scientific Press, pp. 1250026-1-1250026-22, 2012.

2. S. Vipsita and S. K. Rath, "Sequence based protein superfamily classification using computational intelligence techniques : a review," *International Journal of Data Mining and Bioinformatics*, Inderscience. (Accepted for publication on 21-04-2013)

3. S. Vipsita and S. K. Rath, "Two stage approach for protein superfamily classification," *Computational Biology Journal*, vol. 2013, Article ID 898090, Hindawi, pp. 1-12, 2013.

Conferences

1. S. Vipsita and S. K. Rath, "An evolutionary approach for protein classification using feature extraction by artificial neural network," *International Conference on Computer and Communication Technology, 2010 Annual IEEE*, pp. 516-520, 2010.

2. S. Vipsita, B. K. Shee, and S. K. Rath, "An efficient technique for protein classification using feature extraction by artificial neural networks," *in India Conference (INDICON), 2010 Annual IEEE*, pp. 15, 2010.

3. B. K. Shee, S. Vipsita and S. K. Rath, "Protein feature classification using particle swarm optimization and artificial neural networks," *International Conference on Communication, Computing and Security (ACM)*, pp. 313-318, 2011.

4. Swati Vipsita and Santanu Ku. Rath, "Protein superfamily classification using kernel PCA and probabilistic neural network," *in India Conference (INDICON), 2011 Annual IEEE*, pp. 1-6, 2011.