Chapter 7

Conclusion and Directions for Further Research

7.1 An Effective Integrated Intelligent System

The concept of designing machines which could be called intelligent in a sense that matters in real world applications and the subsequent efforts in this direction have resulted in numerous advancements till date. Traditional AI attempted to build intelligent systems using tools such as predicate logic and powerful symbol manipulation techniques. Other research efforts deviated from the symbolic approach and explored the potential for creating intelligent machines by modeling behavior and mechanisms by drawing upon metaphors from biology. This exploration focused on novel computational tools like neural networks, fuzzy logic, approximate reasoning, evolutionary algorithms to cite a few.

Neural networks are designed to model the way in which the brain performs a particular task using a connectionist architecture. They perform useful computations using a distributed and fault tolerant approach and their strength stems from a powerful repertoire of learning algorithms. However, the knowledge learnt by neural networks is difficult to extract and understand. Fuzzy systems, on the other hand, are designed to handle uncertainty that pervades our everyday knowledge and to reason with that knowledge to derive reliable conclusions. The knowledge
base of fuzzy systems, obtained from experts or extracted from data, is much less complex and is represented in the form of if-then rules.

The integration of these two technologies has resulted in powerful intelligent decision making systems that inherit all the strength and advantages of both neural networks and fuzzy systems. In neuro-fuzzy systems—which result from the integration of neural and fuzzy techniques—both neural networks and fuzzy systems complement and co-operate with each other with neural networks fine tuning the knowledge embedded in fuzzy systems and fuzzy logic enabling neural networks to handle fuzzy information thereby making their knowledge base easily interpretable.

Despite their promising capabilities, neuro-fuzzy systems have inherent potential problems concerned with their formulation and development. The architectural design and determination of the optimal network parameters of neuro-fuzzy systems for a given task is crucial in their successful application. One of the techniques used for searching optimal neuro-fuzzy systems are evolutionary algorithms. Even though the search by evolutionary algorithms often yields near optimal solutions, the computationally intensive nature of their search limits the application of evolvable neuro-fuzzy systems to problems involving small data sets. This necessitates a distributed learning methodology for evolutionary neuro-fuzzy systems.

This thesis has focused on the development of an effective integrated neuro-fuzzy system—*Asymmetric Subsethood Product Fuzzy Neural Inference System* (ASuPFuNIS). The development of ASuPFuNIS involved

- study of the less explored utility of asymmetric membership functions to model uncertainty, design of inferencing mechanisms, derivation of gradient descent based learning for the model;

- evaluation of its inferencing abilities on diverse applications such as pattern classification, function approximation and time series prediction, control and medical diagnosis problems;
• exploration of the efficacy of evolutionary algorithms as a learning method;
• formulation of a parallel learning methodology; and
• evaluation of its performance on a real world bioinformatics problem.

In the following sections, we summarize the main contributions of the present research work.

7.2 Asymmetric Subsethood Product Fuzzy Neural Inference System

The Asymmetric Subsethood Product Fuzzy Neural Inference System (ASuPFuNIS), which falls under the class of neuro-fuzzy systems, employs asymmetric Gaussian membership functions to model network weights and feature spreads. Inferencing in ASuPFuNIS is realized by a mutual subsethood based signal transmission method and a product aggregator that works in conjunction with volume defuzzification. In Chapter 3 we described the ASuPFuNIS model with extensive computational expressions underlying its operation.

ASuPFuNIS Architecture

The ASuPFuNIS model directly embeds fuzzy if-then rules of the Mamdani type. Asymmetric Gaussian membership functions, specified by a center, a left spread and a right spread are employed to model fuzzy sets corresponding to the linguistic variables of the embedded fuzzy if-then rules. The architecture of the proposed ASuPFuNIS model as discussed in Section 3.2 has three layers—an input layer, a hidden layer and an output layer. Each node in the input layer represents a domain variable or a feature. Each hidden node represents a fuzzy rule, and input-hidden node connections represent fuzzy rule antecedents. Each output node represents a target variable or a class and hidden-output node connections represent fuzzy rule consequents.
ASuPFuNIS Simultaneously Admits Numeric and Linguistic Inputs

Section 3.3 described the transmission of numeric and linguistic signals through the input nodes of ASuPFuNIS. To handle both these inputs there are two kinds of nodes in the input layer: linguistic nodes and numeric nodes. Linguistic nodes transmit linguistic inputs represented by pre-specified asymmetric Gaussian fuzzy sets without any transformation. Numeric nodes are tunable feature-specific fuzzifiers. They accept numeric inputs and fuzzify them using asymmetric Gaussian fuzzy sets by treating the numeric value as the center of an asymmetric Gaussian membership function with tunable left and right spreads.

ASuPFuNIS uses Mutual Subsethood Based Signal Transmission

Since both the incoming signals and the antecedent weights are fuzzy, signal transmission along antecedent fuzzy weights is handled by calculating their mutual subsethood. Section 3.4 described in detail the mutual subsethood based signal transmission to rule nodes. Comparative studies of the behavior of sup-star and mutual subsethood signal transmission methods were documented in Section 3.5. These studies revealed that the mutual subsethood method has better discriminatory power—it is able to nicely differentiate input vectors that differ from the weight vector over a wider range of spreads than the sup-star composition method.

ASuPFuNIS uses a Product Operator for Aggregation

The input–antecedent weight mutual subsethoods corresponding to connections that fan-in to a rule node, assess the compatibility between the antecedent fuzzy weights and the incoming fuzzy signals. As a measure of this compatibility, ASuPFuNIS aggregates activities at a rule node using products of mutual subsethoods. This product aggregation of activities at rule nodes was described in Section 3.6.
ASuPFuNIS uses Volume Defuzzification

Numeric activation values from the rule nodes are transmitted without any transformation through consequent connections to output nodes. The numeric inference at each output node is determined using volume based centroid defuzzification. The signal computation at the output layer using volume defuzzification was discussed in Section 3.7.

Computational Complexity of ASuPFuNIS Operation

A worst case analysis of the computational complexity of ASuPFuNIS operation was carried out in Section 3.8. The computations involved in both the forward and backward passes of supervised ASuPFuNIS learning were evaluated and tabulated.

ASuPFuNIS is a Universal Approximator

In Section 3.9, this thesis put forth a formal proof that ASuPFuNIS is a universal function approximator. Using the Stone-Weierstrass theorem, it was shown that ASuPFuNIS is capable of approximating any real continuous function on a compact set to arbitrary accuracy.

Learning in ASuPFuNIS

One of the ways in which ASuPFuNIS can be trained is through a supervised learning procedure. It involves repeated presentation of input patterns drawn from the training set and a comparison of the output of the network with the desired values to obtain the error. Network weights are changed on the basis of an error minimization criterion. Once the network is trained to a desired level of error, it is tested by presenting unseen patterns. A comprehensive set of equations was derived in Section 3.10 for the weight update procedure involved in supervised learning. Also, the potential of applying evolutionary algorithms to train ASuPFuNIS was investigated in Chapter 5.
7.3 Benchmarking ASuPFuNIS on Various Applications

The ASuPFuNIS model finds application in diverse fields such as pattern classification, function approximation, time series prediction, control and medical diagnosis. In Chapter 4, the high performance and the generalization abilities of the ASuPFuNIS model were benchmarked on the aforesaid applications.

7.3.1 ASuPFuNIS Applications in Pattern Classification

The classification capabilities of ASuPFuNIS were demonstrated on three benchmark classification problems—Ripley's synthetic data classification, Iris data classification and Telugu vowel classification. In all three applications, ASuPFuNIS was shown to perform excellently as a classifier both in terms of performance and architectural economy.

Ripley's Synthetic Data Classification

Section 4.4 dealt with the classification of Ripley's synthetic two class data by ASuPFuNIS. Ripley's data comprises patterns having two features divided into two classes. The training set consists of 250 such patterns with 125 patterns in each class and the test set consists of 1000 such patterns with 500 patterns in each class. The class distributions have been chosen to allow a Bayesian classifier test error rate of 8%. Simulations were conducted with 2, 3 and 4 rule ASuPFuNIS networks, and the test errors obtained by these networks were respectively 7.8%, 7.5% and 7.4%. These results are much better in terms of performance (classification accuracy) and architectural economy when compared with numerous other models such as subssethood product fuzzy neural inference system (SuPFuNIS), multilayer perceptron (MLP), learning vector quantization (LVQ), constrained topological mapping (CTM) and linear discriminant.
7.3. Benchmarking ASuPFuNIS on Various Applications

Iris Data Classification

This problem, discussed in Section 4.5, involves classification of three subspecies of the Iris flower on the basis of its four feature measurements. The input pattern set comprises 150 four-dimensional patterns. Once trained with these 150 patterns, the test set which again comprise all 150 patterns is used to compute resubstitution error. ASuPFuNIS obtained zero resubstitution error with just three rules. In comparison, for 3 rules/prototypes, other techniques like learning vector quantization produced 14 misclassifications; genetic algorithm produced 4 misclassifications and random search produced 2 misclassifications. Other soft computing models like fuzzy genetic neural system (FuGeNeSys) and SuPFuNIS obtained 100% resubstitution accuracy with 5 rules, a neuro-fuzzy classifier NEFCLASS with 7 rules obtained 96.7% accuracy, a recurrent evolving fuzzy neural network (ReFuNN) and an evolving fuzzy neural network (EFuNN) with respectively 9 and 17 rules obtained 95.3%. A multilayer perceptron obtained zero resubstitution error with 4-6-6-3 architecture.

Telugu Vowel Classification

The Telugu vowel classification problem discussed in Section 4.6 involves classification of 871 Indian Telugu vowel sounds into one of 6 vowel classes viz. \( \partial \), \( a \), \( i \), \( u \), \( e \) and \( o \). The training set was generated by randomly choosing 10% of samples of each class while remaining patterns were retained as the test set. Experiments were performed on 12 such training and test sets. ASuPFuNIS obtained a correct recognition rate of 83.80% with 3 rules, 84.18% with 4 rules and 84.31% with 6 rules. In comparison, SuPFuNIS obtained correct recognition rates of 80.87% with 6 rules and 82.87% with 10 rules. The three variants of Pal and Mitra’s neuro-fuzzy model are reported to obtain correct recognition rates of 76.8% with 20 rules, 80.1% with 22 rules and 84.2% with three hidden layers and 10 nodes in each layer.
7.3. Benchmarking ASuPFuNIS on Various Applications

7.3.2 ASuPFuNIS Applications in Function Approximation and Time Series Prediction

The approximation and prediction capabilities of ASuPFuNIS were tested on two benchmark problems: Narazaki-Ralescu function approximation problem; and a complex non-linear dynamical time series (Mackey Glass Time Series) prediction problem.

Narazaki-Ralescu Function Approximation Problem

Section 4.7 dealt with the approximation of a single input single output (SISO) function: \( y(x) = 0.2 + 0.8(x + 0.7 \sin(2\pi x)) \), \( 0 \leq x \leq 1 \) suggested by Narazaki and Ralescu. Twenty-one training patterns were generated at intervals of 0.05 and 101 test data were generated at intervals of 0.01 both in the range (0, 1). The performance indices \( J_1 \) (learning capacity) and \( J_2 \) (generalization capacity) were used to evaluate the approximation performance of ASuPFuNIS. A 2 rule ASuPFuNIS yielded \( J_1 = 0.0979\% \) and \( J_2 = 0.0916\% \), while a 3 rule network yielded \( J_1 = 0.0357\% \) and \( J_2 = 0.0391\% \). By contrast, a genetic fuzzy rule extractor GEFREX yielded \( J_1 = 0.103\% \) and \( J_2 = 0.090\% \) and a 3 rule SuPFuNIS resulted in \( J_1 = 0.3061\% \) and \( J_2 = 0.3310\% \) while a 5 rule SuPFuNIS obtained \( J_1 = 0.0957\% \) and \( J_2 = 0.0972\% \). With three rules ASuPFuNIS outperformed both GEFREX and SuPFuNIS.

Mackey-Glass Time Series Prediction

The Mackey-Glass time series is generated by the delay differential equation:
\[
\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t)
\]
This problem, discussed in Section 4.8, involves predicting a future value \( x(t + \Delta t) \) (\( \Delta t \) being the prediction time step) based on a set of values of \( x(t) \) at certain times less than \( t \). From the Mackey-Glass time series, 1000 input-output data pairs were extracted. The first 500 input-output data tuples were used as the training set and the second 500 input-output data tuples were employed as the test set.
With 5 and 10 rules, ASuPFuNIS yielded a normalized root mean squared error (NRMSE) of 0.0148 and 0.0080 respectively on the test set. A non-singleton fuzzy logic system (NSFLS) employing the symmetric Gaussian input and antecedent fuzzy sets with product inference was also implemented and tested on the same problem. NSFLS obtained an NRMSE of 0.0107. With 5 and 10 rules, SuPFuNIS yielded NRMSEs of 0.016 and 0.014 respectively. An adaptive neuro fuzzy inference system (ANFIS) and GEFREX outperformed all other models with NRMSE values 0.0074 and 0.0061 respectively. However, ANFIS has the drawback that it has less interpretability in terms of learned information; and the implementation of GEFREX is difficult.

### 7.3.3 ASuPFuNIS Applications in Control

Section 4.9 dealt with the truck backer-upper control problem with the objective of backing up a truck to a loading dock. The truck position is determined by three state variables $\phi$, $x$, and $y$ where $\phi$ is the angle of the truck with the horizontal, $x$ and $y$ are the co-ordinates in the space. The control of the truck is the steering angle $\theta$. The truck moves backward by a fixed unit distance every stage. We designed a neuro-fuzzy control system using the ASuPFuNIS model, whose inputs are $x \in [0, 20]$ and $\phi \in [-90^\circ, 270^\circ]$ and whose output is $\theta \in [-40^\circ, 40^\circ]$ such that the final state (loading dock) was $(x_f, \phi_f) = (10, 90^\circ)$. The training data comprised 238 pairs which were accumulated from 14 sequences of desired $(x, \phi; \theta)$ values. In the present problem, the performance of a controller is considered good if a proper balance is maintained between the type of trajectory and docking accuracy. Several initial states $(x, \phi_0)$, were used to test the performance of the controller. Simulation results showed that 2 and 3 rule ASuPFuNIS networks performed very well in comparison to a 5 rule SuPFuNIS network, as well as Kong and Kosko's fuzzy controller which used 35 linguistic rules and Wang and Mendel's controller which used 27 rules.
7.4. Evolutionary Learning for ASuPFuNIS

7.3.4 ASuPFuNIS Application in Medical Diagnosis Involving Mixed Numeric-Linguistic Data

A medical diagnosis problem—hepatitis diagnosis—discussed in Section 4.10 demonstrated the ease with which ASuPFuNIS can simultaneously handle numeric as well as linguistic information and simultaneously be robust against random variations in the data. This problem requires the network to classify hepatitis patients into two classes Live and Die on the basis of features which are both numeric and linguistic. The hepatitis data set has 155 patterns of 19 input features with a number of missing values. The class distribution of the data is such that 32 patterns belong to class Die and the remaining 123 patterns belong to class Live. As the data set is incomplete, two data sets were constructed using average values of missing features calculated on a class-wise basis from the 80 originally complete data. Experiments were done on five train-test combinations of both data sets. ASuPFuNIS had a high test classification accuracy ranging from 91.67% to 100% with 2 to 4 rules on both data sets.

All the aforesaid benchmark applications dealt with in Chapter 4 prove that ASuPFuNIS has a superior performance and generalization ability with architectural economy solving all the problems with a fewer number of rules and with a much higher performance. Also, the ability of the model to handle numeric and linguistic data seamlessly and to be robust against random variations in the data were demonstrated.

7.4 Evolutionary Learning for ASuPFuNIS

Chapter 5 explores the efficacy of evolutionary algorithms as a learning method for the ASuPFuNIS model. Preliminary experiments were conducted with binary and real-coded genetic algorithms (GA's) to evolve SuPFuNIS network parameters. A novel evolutionary algorithm—Differential Evolution (DE)—was employed
7.4. Evolutionary Learning for ASuPFuNIS
to learn the trainable parameters and antecedents of ASuPFuNIS. A natural outcome of the encoding technique employed is the simultaneous learning of significant features as well as the number of rule nodes of ASuPFuNIS, which was also demonstrated.

7.4.1 Differential Evolution Learning in ASuPFuNIS
Section 5.2 briefly described the preliminary experiments with binary and real-coded genetic algorithms employed to learn the trainable parameters of the SuPFuNIS model developed earlier in our laboratory. The experiments revealed the issues involved in the parameter settings of the genetic algorithms. To alleviate these issues with GA’s a simple and a novel search method—Differential Evolution (DE)—has been employed to learn the network parameters, antecedent connectivity pattern of ASuPFuNIS as well the significant features of the undertaken task. Section 5.3 described the various operators and operations involved in the DE algorithm as well as the genetic coding scheme employed.

Differential Evolution is a novel population based parallel search method with a new scheme for generating trial vectors. The genetic coding employed in the present work used both real and binary parts with the real part encoding the fuzzy weights of ASuPFuNIS, and the binary part encoding antecedent connection information. The variant of DE employed in this thesis perturbs the best real vector of current generation with weighted difference between two random population members (DE/best/1). The weighted difference and trial vector generation operations on the enable bits were modeled by a customized novel dissimilarity based bit flipping operator. Crossover in the real part involved shuffling between a trial vector and a predetermined population member while the binary part employed a uniform crossover operation.

The effectiveness of DE learning for ASuPFuNIS was tested on two different function approximation benchmark problems—Narazaki-Ralescu function approximation problem and the HANG problem—a chemical plant control problem and on the Iris data classification problem.
Narazaki-Ralescu Function Approximation Problem

In Section 5.4, DE was employed to learn the ASuPFuNIS network parameters for approximating the Narazaki-Ralescu function. With 2 and 3 rules ASuPFuNIS yielded $J_2 = 0.0522\%$ and $J_2 = 0.0135\%$ respectively. This should be compared against the 2 and 3 rule ASuPFuNIS networks, with gradient learning, which yielded $J_2 = 0.0916\%$ and $J_2 = 0.0390\%$ respectively. This marks a significant improvement in the performance of ASuPFuNIS network using DE learning as compared to the gradient learning based model.

HANG Problem

This problem, as discussed in Section 5.5, involves approximation of a non-linear function: $y = (1 + x_1^2 + x_2^{-1.5})^2$, $1 \leq x_1, x_2 \leq 5$. The HANG data set comprises 50 input-output data points generated by randomly picking 50 pairs of points $x_1, x_2$. In the parameter learning experiment, a 3 rule ASuPFuNIS network resulted in a root mean squared error (RMSE) of 0.0432 and a performance index (PI) of 0.0024. In comparison, the flexible neuro-fuzzy inference system (FLEXNFIS) yielded an RMSE = 0.0731 (PI not reported in [246]) and Lin and Cunningham's model gave a PI of 0.0035 (RMSE not reported in [169]). In the simultaneous antecedent connectivity and parameter learning experiment, the DE pruned both antecedent connections to the third rule thereby pruning a rule to get a 2 rule ASuPFuNIS network from an initial 3 rule network, with a final PI of 0.0076.

To demonstrate the feature selection capability of DE learning, two random variables $x_3$ and $x_4$, in the range [1, 5] were added as dummy inputs to the system. The 3-rule ASuPFuNIS network with DE learning successfully pruned the indifferent features i.e., $x_3$ and $x_4$ and obtained a high performance index measure of 0.0039. Both Sugeno and Yasukawa's fuzzy model and Chakraborty and Pal's neuro-fuzzy model obtained a PI of 0.01 (RMSE not reported in [260]). To facilitate the comparison with aforementioned models the performance of ASuPFuNIS
on this problem has been reported both in terms of performance index (PI) and root mean-squared error (RMSE).

**Chemical Plant Control Problem**

Section 5.6 described this control problem which involves the control of a chemical plant for producing a polymer by polymerization of some monomers. The problem contains 70 data points obtained from an actual plant operation each comprising five input features that a human operator may refer for control and one output which is the setpoint of monomer flow rate. Of the five input features, feature four and five referring to local temperatures inside plant do not significantly contribute to the output.

For this control problem, DE learning was employed to learn the significant features as well as network parameters for ASuPFuNIS. The DE employed a sum of sum-of-squared errors (SSE) and a penalty term as fitness function and after a run of 4700 DE generations successfully pruned the insignificant features (i.e., 4 and 5) and obtained a performance index of 0.0021. In addition, DE learning also pruned three antecedent connections thereby evolving an optimal network. In comparison, Lin and Cunningham’s model [169] and Chakraborty and Pal’s [38] model obtained performance indices of 0.0022 and 0.0021 respectively. The former employed 7 rules and the latter employed 32 rules to achieve the reported performance. However, ASuPFuNIS employed just 3 rules to achieve a comparable performance. This experiment demonstrates the efficient learning capability of DE as well as the architectural economy and inferencing capabilities of ASuPFuNIS.

**Iris Data Classification Problem**

The Iris data classification problem involves classification of three species of Iris flower on the basis of four features viz., sepal length, sepal width, petal length and petal width. However it is well known that consideration of only features 3 and 4 (petal length and petal width) is sufficient to solve this classification problem. To learn these significant features as well as the ASuPFuNIS network parameters, DE
was employed. Section 5.7 described the experiment details and simulation results of DE run on this problem. The DE run employed a fitness function involving sum-of-squared errors, a penalty term and misclassification information. After a run of 1389 DE generations, ASuPFuNIS successfully identified the significant features and achieved a resubstitution error of 4 on this classification problem while pruning one antecedent connection. This performance of ASuPFuNIS is comparable with that of Chakraborty and Pal’s model which also obtained a resubstitution error of 4 while pruning features 1 and 2. While the latter model employed 5 rules, ASuPFuNIS employed 3 rules to achieve this performance. This experiment reiterates the success of evolutionary algorithms as efficient learning methods for ASuPFuNIS as against the conventional gradient based learning method.

7.5 A Parallel Framework for Implementing ASuPFuNIS

Chapter 6 focused on preliminary efforts to develop a parallel framework for implementing ASuPFuNIS with significant speedups in the evolutionary learning process. The parallel model and implementation adopted to parallelize ASuPFuNIS learning involved distributing the fitness evaluation operation of the evolutionary learning to multiple computers.

7.5.1 Master-Slave Parallel DE Using PVM

We employed a panmictic master-slave model to parallelize the differential evolution learning of ASuPFuNIS. This methodology discussed in Section 6.2 evaluates in parallel the individuals of the population while still using selection, mutation and recombination operations which are performed in the main processor that guides the basic DE algorithm. There is a master process that spawns a number of slave processes. The master process then distributes the population to be evaluated and every slave process starts to work and send back the evaluation
results to the master process. After having completed specified number of DE generations, the master process collects results from the slave processes and kills them. The master-slave parallel DE is implemented using message passing model of communication, in which processes in the same or on physically different processors communicate by sending messages to each other through a communication medium, such as a standard or specialized network connection. Section 6.3 described the message passing model of communication and the tools to implement it. In this thesis, the message passing model of communication was realized with Parallel Virtual Machine (PVM). The details regarding the setup of PVM and parallel implementation of ASuPFuNIS using PVM were described in Section 6.4.

Narazaki-Ralescu Function Approximation Problem: A Case Study

As a case study, Section 6.5 described the effect of parallelizing ASuPFuNIS learning on the Narazaki-Ralescu function approximation problem. For parallelization experiments four computers were used. There were three Pentium III (1 GHz) systems used as slaves and one Pentium III (0.8 GHz) system used as master. The OS was Red Hat Linux 7.2. Simulations were conducted with parallel DE using 1, 2 and all three slaves. While serial DE took 33.48 minutes to evolve the network parameters, parallel DE with single, two and three slaves took respectively 8.45 minutes, 5.16 minutes and 4.16 minutes. The speedup achieved in each case was super-linear. The fact that the methodology adopted could achieve super-linear speedup, for a problem not so computationally intensive, suggests that for problems having computationally costly objective functions this methodology would produce very high super-linear speedups.

7.6 Parallel ASuPFuNIS Applications

Section 6.6 presented an application of Parallel ASuPFuNIS on a bioinformatic problem—classification of highly similar cancerous tissues using gene expression information.
7.6.1 Gene Expression Classification of Cancerous Tissue

The bioinformatics data set, employed in the work, is a collection of 7129 gene expression profiles constructed from 72 patients suffering from either acute lymphoblastic leukemia or acute myeloid leukemia. Out of 72 leukemia samples, 38 samples form the training set and the remaining 34 samples form the testing set. The task is to classify the samples into either lymphoblastic or myeloid case based on the gene expression numbers. A subset of 50 informative genes (chosen by Golub et al.), out of 7129, that correlates well with the two classes of leukemias was adopted in the present work.

Since the number of network parameters is 412 for a 2-rule ASuPFuNIS and the DE population size is 4120 for this classification problem, the simulation was conducted in a parallel framework with three slaves. After a run of 324 DE generations, ASuPFuNIS correctly classified all the 38 training samples and gave 2 misclassification errors on the test samples. To qualitatively analyse the classification performance of ASuPFuNIS, a confidence factor (CF) was calculated. Out of 38 training samples 36 samples had a classification confidence greater than 0.99 with 2 samples above 0.4. In the case of the test samples, all the correctly classified 32 samples had a classification confidence above 0.8 which is considerably high. Golub et al. performed hold-one-out cross validation tests and got 36 out of 38 training samples correctly classified and 2 were disregarded as uncertain. In the case of the test set, 29 out of 34 samples were correctly classified and for the remaining 5 samples classification was uncertain. Furey et al. used a Support Vector Machine (SVM) based classification along with the hold-one-out cross validation that classified correctly all the 38 training samples while on the test set it has been reported that 30 to 32 of the 34 samples were classified correctly. The performance of ASuPFuNIS is comparable with the work of Golub and Furey.
7.7 Avenues for Further Research

The asymmetry in Gaussian membership functions in ASuPFuNIS has provided the model with a high level of flexibility and computational strength as has been demonstrated by its superior performance in all the benchmarking applications discussed. However, often since the human language is too vague to decide the asymmetry in membership functions, the correct interpretation of the asymmetric Gaussian membership functions in a trained ASuPFuNIS network into if-then rule form is an issue to be analysed and investigated.

Despite the fact that the structural evolution of the ASuPFuNIS model using evolutionary algorithms has been attempted in the present research work, a comprehensive methodology needs to be formulated for the complete evolution of ASuPFuNIS. Unlike the implicit learning of the number of rule nodes as demonstrated in the present work, the methodology should learn or simultaneously evolve the

- optimal number of rule nodes for a given task,
- optimal antecedent connectivity patterns,
- significant features, and
- optimal set of network parameters.

This would enable the model to evolve to somewhat optimal architecture depending on the task at hand.

Regarding the parallel implementation of ASuPFuNIS a simple synchronous master-slave methodology, albeit with an excellent performance, has been adopted in the present work. More sophisticated parallel learning methodologies like distributed and cellular methods, less explored for the neuro-fuzzy systems, need to be investigated. To cite an example, in the island model implementation of distributed neuro-fuzzy-evolutionary learning, while each computer performs individual learning on a set of population of neuro-fuzzy networks focusing on a
subspace of the search space, they also occasionally exchange their best found networks among themselves. This individual as well as co-operative search is one of the effective distributed learning methodologies that has the potential not only to speed up the learning process but also to improve the solution quality. The parallel implementation of ASuPFuNIS learning can be formulated with such powerful distributed learning methodologies.

In the present work, the design of ASuPFuNIS network for a particular task has been done either heuristically or by partly using evolutionary algorithm based search. Techniques that could extract the crude domain knowledge can be employed to initialize the ASuPFuNIS network design. This would impart the domain knowledge in network architecture to make it much more efficient than the conventional heuristically designed ASuPFuNIS networks. Rough-set theory, an important tool to handle uncertainty, is a powerful approach to extract domain knowledge. Rough set theoretic concepts can be integrated with the present ASuPFuNIS to achieve both knowledge discovery at the level of data acquisition and the structuring of the network. The issue of whether support vector learning (that aims at minimizing an upper bound on the generalization error of a model rather than the error over the data set), can employed to provide a framework for ASuPFuNIS structural design can also be investigated.

The existing ASuPFuNIS model embeds Mamdani type fuzzy rules. Efforts can be made to incorporate functional consequent weights so as to embed TSK type rules in the model. The mathematical bounds on the function approximation and control capabilities of such a network can also be investigated.

The hardware implementation of the ASuPFuNIS model can also be studied. This would enable the on-line evolution of the model on the hardware level depending on the functional requirements of the task.