Evolutionary Modular MLP with Rough Sets and ID3 Algorithm for Staging of Cervical Cancer

Pabitra Mitra, Sushmita Mitra and Sankar K. Pal

Indian Statistical Institute, Calcutta, India

This paper describes a way of designing a hybrid system for detecting the different stages of cervical cancer. Hybridisation includes the evolution of knowledge-based subnetwork modules with GAs using rough set theory and the ID3 algorithm. Crude subnetworks for each module are initially obtained via rough set theory and the ID3 algorithm. These subnetworks are then combined, and the final network is evolved using genetic algorithms. The evolution uses a restricted mutation operator which utilises the knowledge of the modular structure, already generated, for faster convergence. The GA tunes the network weights and structure simultaneously. The aforesaid integration enhances the performance in terms of classification score, network size and training time, as compared to the conventional MLP. This methodology also helps in imposing a structure on the weights, which results in a network more suitable for rule extraction.

Keywords: Cervical cancer; Classification; Genetic algorithms; Knowledge-based systems; Modular neural networks; Rule extraction

1. Introduction

The worldwide occurrence of cancer of the cervix cases shows [1] that only 20% of these cases occur in developed nations, while 80% are found in developing countries, including India [2]. In India, cancer of the uterine cervix is the most frequent malignancy observed in females, as per the reports of the different cancer registries published in the I.C.M.R annual report [3].

The medical records of the Chittaranjan National Cancer Institute (CNCI), Calcutta, for the last five years, show that the commonest malignancy observed in females is cancer of the cervix, which comprises about 40% of the total female cases diagnosed. Cervical cancer has a more or less well defined treatment modality. Treatments available for this particular type of malignancy are surgery, radiotherapy and chemotherapy, or a combination of all of these, depending on the stage of the disease at the time of diagnosis, and also on the physical condition of the patient during treatment.

Staging is a process that uses information learnt about cancer through diagnostic processes, such as the size of the tumor, how deeply the tumor has invaded tissues at the site of origin, the extent of any invasion into surrounding organs, and the extent of metastasis (spread) to lymph nodes or distant organs. This is a very important process, because proper staging is the most important factor in selecting the right treatment plan. Cervical cancer is most frequently staged using the FIGO (International Federation of Gynaecology and Obstetrics) system of staging. This system classifies the disease in Stages I through IV.

With the advent of various new modalities of treatment in the field of cancer, the decision making towards a particular treatment regime to be adopted for each individual patient has become a complex process, and should keep pace with advancements in medical science. More often, there is a large amount of information to be processed, much of which is quantifiable. Intuitive thought processes involve rapid unconscious data processing, and combine the available information by the law of averages, and consequently, have a low intra- and inter-

Correspondence and offprint requests to: P. Mitra, Machine Intelligence Unit, Indian Statistical Institute, Calcutta 700035, India. Email: {pabitra_r,sushmita,sankar}@isical.ac.in

person consistency. So from the point of intuitive decision making, the clinician of today should move towards analytic decision making which, though typically slow, is conscious and consistent, and clearly spells out the basis of the decisions.

In the field of oncology, decision making is not only restricted to finding out the correct diagnosis and planning of the proper treatment modality, but also to take cognizance of factors like the patient's socio-economic background, his or her ability to pursue a prolonged and expensive treatment program, and also whether the ill-effects of the treatment modalities will outweigh its efficacy. Unlike other diseases, comprehensive cancer treatment involves not only the above-mentioned treatment interventions at the onset of the disease, but a life-time follow-up information on each patient is essential to prevent recurrence, and to calculate the disease-free survival necessary to evaluate any treatment efficacy. If a computerised program could be developed taking all these factors into consideration, that would help in the analytical decision making towards treatment and other related parameters; it would go a long way to make the task of a practicing oncologist much easier.

The objective of this paper is to design a medical decision support system for cancer management, using a knowledge-based network in combination with rough set theory and Genetic Algorithms (GA) in soft computing paradigm. The proposed system is able to exploit the parallelism, self-learning and fault tolerance characteristics of Artificial Neural Network (ANN) models, knowledge encoding capabilities of rough set theory, and the adaptive, parallel and robust searching characteristics of GAs. The model is built on data of cancer of the uterine cervix for detecting its various stages.

Artificial Neural Networks (ANNs) generally consider a fixed topology of neurons connected by links in a pre-defined manner. These connection weights are usually initialised by small random values. Recently, there have been some attempts at improving the efficiency of neural computation by using knowledge-based nets. This helps in reducing the searching space and time while the network traces the optimal solution. Knowledge-based networks [4–6] constitute a special class of ANNs that consider crude domain knowledge to generate the initial network architecture, which is later refined in the presence of training data. Such a model has the capability of outperforming a standard multilayer perceptron (MLP), as well as other related algorithms, including those based on symbolic and numerical ones [4,5]. Recently, the theory of rough sets has been used to generate knowledge-based networks.

The theory of *rough sets* [7,8] has emerged as a major mathematical tool for managing uncertainty that arises from granularity in the domain of discourse, i.e. from the indiscernibility between objects in a set. The primary role of rough sets here is in managing uncertainty and extracting domain knowledge [9].

The ID3 approach [10,11] to pattern recognition and classification consists of a procedure for synthesizing an efficient discrimination tree for classifying patterns that have non-numeric attributes or feature values. The discrimination tree can also be expressed in the form of a body of rules and, because of this, ID3 is also often thought of as an inductive inference procedure for machine learning or rule acquisition. These rules can also be encoded to generate a knowledge-based network.

A recent trend in neural network design for large scale problems is to split the original task into simpler subtasks, and use a subnetwork module for each of the subtasks [12]. The popular methods available for decomposition include the Local Model Network (LMN) [13] and the CALM (Categorising And Learning Module) model [12]. It has been shown that by combining the output of several subnetworks in an ensemble, one can improve the generalisation ability over that of a single large network [14].

Genetic Algorithms (GAs) are randomised search and optimisation techniques guided by the principles of evolution and natural genetics [15]. They are efficient, adaptive and robust search processes, producing near-optimal solutions and having a large amount of implicit parallelism. Therefore, the application of genetic algorithms for solving certain problems of pattern recognition (which need optimisation of computation requirements, and a robust, fast and close approximate solution) appears to be appropriate and natural [16]. Many researchers have combined genetic algorithms with neural networks for building more powerful adaptive systems. Here one simultaneously evolves the optimal set of weight values and thresholds along with the network topology and learning parameters [17,18].

The connection weights of these evolved modular knowledge-based networks can be used for extracting refined rules for the problem domain [19]. Such models help in minimising human interaction and associated inherent bias during the phase of knowledge-base formation, and also reduce the possibility of generating contradictory rules. The extracted rules help in alleviating the knowledge acquistion *bottleneck*, refining the initial domain knowledge, and providing reasoning and explanation facilities. One realises that, especially in the medical

domain, the final responsibility for any diagnostic decision always has to be accepted by the medical practitioner. So the doctor may want to verify the justification behind the decision reached, based on his/her expertise. This requires the system to be able to explain its mode of reasoning for any inferred decision/recommendation, preferably in rule form, to convince the user that its reasoning is correct. All these aspects serve to demonstrate the utility of automated rule extraction in a medical decision support system.

In the present paper, an evolutionary strategy is suggested for designing a modular knowledge-based network using both rough set theory and the ID3 algorithm. The evolutionary training algorithm generates the weight values for a parsimonious network. Rough set theory and the ID3 algorithm are used to obtain the sets of probable knowledge-based subnetworks which form the initial population of the GA. These modules are then integrated and evolved with a *restricted* mutation operator that helps preserve extracted localised rule structures as potential solutions. This type of 'divide and conquer' strategy accelerates the training significantly, as compared to the training of the entire network. A restricted mutation operator is implemented, which utilises the knowledge of the modular structure evolved to achieve faster convergence. Classification performance of the models for different stages is compared with that of the conventional MLP. The rules extracted are also verified by oncologists.

2. Knowledge-based MLP

The output of a neuron in any layer (h) other than the input layer ($h = 0$) is given as

$$y_j^h = \frac{1}{1 + \exp(-\Sigma_i y_i^{h-1} w_{ji}^{h-1})} \tag{1}$$

where y_i^{h-1} is the state of the ith neuron in the preceding $(h-1)$th layer and w_{ji}^{h-1} is the weight of the connection from the ith neuron in layer $h-1$ to the jth neuron in layer h. For nodes in the input layer, y_j^0 corresponds to the jth component of the input vector. Note that $x_j^h = \Sigma_i y_i^{h-1} w_{ji}^{h-1}$.

Knowledge is extracted using two methods: (1) rough set theoretic concepts; and (2) the ID3 algorithm. The extracted crude domain knowledge is encoded among the connection weights of an MLP. This helps one to automatically generate an appropriate network architecture in terms of hidden nodes and links. The methods model arbitrary decision regions with multiple object representatives. The knowledge encoding algorithms are radically different from existing models [4,5].

2.1. Rough MLP

The formulation of a Rough MLP [20] is described in this section. The feature space gives the condition attributes and the output classes the decision attributes, so as to result in a decision table. This table may be transformed, keeping the complexity of the network to be constructed in mind. Rules are then generated from the (transformed) table by computing relative reducts. The dependency factors of these rules are used to encode the initial connection weights of the resultant knowledge-based network.

2.1.1. Rule Generation. Let $\mathcal{S} = <U,A>$ be a decision table, with C and $D = \{d_1, \ldots, d_l\}$ its sets of condition and decision attributes, respectively. Divide the decision table $\mathcal{S} = <U,A>$ into l tables $\mathcal{S}_i = <U_i,A_i>$, $i = 1, \ldots,l$, corresponding to the l decision attributes $d_1, \ldots,d_l$, where

$$U = U_1 \cup \ldots \cup U_l \quad \text{and} \quad A_i = C \cup \{d_i\}$$

The size of each $\mathcal{S}_i$ $(i = 1, \ldots,l)$ is first reduced with the help of a threshold on the number of occurrences of the same pattern of attribute values. This will be elicited in the sequel. Let the reduced decision table be denoted by $\mathcal{T}_i$, and $\{x_{i1}, \ldots,x_{ip}\}$ be the set of those objects of U_i that occur in $\mathcal{T}_i, i = 1, \ldots,l$.

Now for each d_i-reduct $B = \{b_1, \ldots,b_k\}$ (say), a discernibility matrix (denoted $\mathbf{M}_{d_i}(B)$) from the d_i-discernibility matrix is defined as follows:

$$c_{ij} = \{\alpha \in B : a(x_i) \neq a(x_j)\} \tag{2}$$

for $i, j = 1, \ldots,n$.

For each object $x_j \in x_{i_1}, \ldots,x_{i_p}$, the discernibility function $f_{d_i}^{x_j}$ is defined as

$$f_{d_i}^{x_j} = \wedge \{\vee(c_{ij}) : 1 \leq i,j \leq n, j < i, c_{ij} \neq \varnothing\} \tag{3}$$

Then $f_{d_i}^{x_j}$ is brought to its conjunctive normal form (c.n.f). One thus obtains a dependency rule r_i, *viz.* $P_i \leftarrow d_i$, where P_i is the disjunctive normal form (d.n.f) of $f_{d_i}^{x_j}, j \in i_1, \ldots,i_p$.

The dependency factor df_i for r_i is given by

$$df_i = \frac{card(POS_i(d_i))}{card(U_i)} \tag{4}$$

where $POS_i(d_i) = \cup_{X \in I_{d_i}} l_i(X)$, $l_i(X)$ is the lower approximation of X with respect to I_i. In this case, $df_i = 1$ [20].

2.1.2. Knowledge Encoding. Consider the case of class c_k in the l-class problem domain. As the method considers multiple objects in a class, a separate $n_k \times n$-dimensional attribute-value decision table is generated for each class c_k (where n_k indicates the number of objects in c_k).

Let there be m sets $O_1, \ldots, O_m$ of objects in the table having identical attribute values, and $card(O_i) = n_{k_i}$, $i = 1, \ldots, m$, such that $n_{k_1} \geq \ldots \geq n_{k_m}$ and $\Sigma_{i=1}^{m} n_{k_i} = n_k$. The attribute-value table can now be represented as an $m \times n$ array. Let $n_{k'_1}, n_{k'_2}, \ldots, n_{k'_m}$ denote the distinct elements among $n_{k_1}, \ldots, n_{k_m}$ such that $n_{k'_1} > n_{k'_2} > \ldots > n_{k'_m}$. Let a heuristic threshold function be defined as

$$Tr = \left[\frac{\Sigma_{i=1}^{m} \dfrac{1}{n_{k'_i} - n_{k'_{i+1}}}}{Th} \right] \tag{5}$$

where $\vee(c_{ij})$ is the disjunction of all members of c_{ij}, so that all entries having frequency less than Tr are eliminated from the table, resulting in the reduced attribute-value table. Note that the main motive of introducing this threshold function lies in reducing the size of the resulting network. One attempts to eliminate noisy pattern representatives (having lower values of n_{k_j}) from the reduced attribute-value table.

While designing the initial structure of the network, the union of the rules of the l classes is considered. The input layer consists of n attribute values, while the output layer is represented by l classes. The hidden layer nodes model the conjuncts in the antecedent part of a rule. The output layer nodes model the disjuncts. For each conjunct, corresponding to one output class (one dependency rule), one hidden node is dedicated. Only those input attributes that appear in this conjunct are connected to the appropriate hidden node, which in turn is connected to the corresponding output node. Each disjunct is modelled at the output layer by joining the corresponding hidden nodes.

Let the dependency factor for a particular dependency rule for class c_k be 1. The weight w_{ki}^{1} between a hidden node i and output node k is set at $\frac{1}{fac} + \epsilon$, where fac refers to the number of disjuncts in the antecedent of the rule, and ϵ is a small random number taken to destroy any symmetry among the weights. Note that $fac \geq 1$, and each hidden node is connected to only one output node. The weight $w_{ia_j}^{0}$ between an attribute a_j and hidden node i is set to $\frac{1}{facd} + \epsilon$, such that $facd$ is the number of attributes connected by the corresponding conjunct. Again, $facd \geq 1$. Thus, for an l-class problem domain, there are at least l hidden nodes. All other possible connections in the resulting MLP are set as small random numbers. It should be mentioned that the number of hidden nodes is determined from the dependency rules.

2.2. ID3 Algorithm

ID3 is effective when there is a body of data consisting of a large number of patterns, each of which is made up of a long list of nonumeric feature/attribute values. The class membership of some of these patterns are known. The task is to examine the bewilderingly large body of data, and to find out what minimum combination of feature values suffices to determine class membership.

In the ID3 approach [10], we make use of labelled examples and determine how features might be examined in sequence until all the labelled examples have been classified correctly. We might find, for example, that only a very small fraction of the features need be used for classification purposes. If this result obtained for the labelled examples, is also representative of the much larger ensemble of patterns comprising the original body of data, then a very large gain will have been achieved through use of ID3. In addition, the class membership depends upon certain combinations of feature values, as discovered by the discrimination tree, might also provide insight into the basic mechanism that determine the processes being examined.

2.2.1. Model. ID3 uses an information-theoretic approach. The procedure is that at any point we examine the feature that provides the greatest gain in information or, equivalently, the greatest decrease in entropy. Entropy is defined as $-p \log_2 p$, where probability p is determined on the basis of frequency of occurrence.

The general case is that of N labelled patterns partitioned into sets of patterns belonging to classes $c_i, i = 1, 2, 3, \ldots, l$. The population in class c_i is n_i. Each pattern has n features, and each feature has J (≥ 2) values. The ID3 prescription for synthesizing an efficient decision tree can be stated as follows [10]:

Step 1. Calculate initial value of entropy

$$Entropy(I) = \sum_{i=1}^{I} -(n_i/N)\log_2(n_i/N) \tag{6}$$

$$= \sum_{i=1}^{I} -p_i \log_2 p_i \tag{7}$$

Step 2. Select that feature which results in the maximum decrease in entropy (gain in information), to serve as the root node of the decision tree.
Step 3. Build the next level of the decision tree providing the greatest decrease in entropy.
Step 4. Repeat Steps 1 through 3.

Continue the procedure until all subpopulations are of a single class and the system entropy is zero. At this stage, one obtains a set of leaf nodes (subpopulations) of the decision tree, where the patterns are of a single class. There can be some nodes which cannot be resolved any further. Now we describe the knowledge encoding algorithm using the decision tree generated by ID3. Let us consider the leaf nodes only. The path from the root to a leaf can be traversed to generate the rule corresponding to a pattern from that class. In this manner, one obtains a set of rules for all the pattern classes, in the form of intersection of the features/attributes encountered along the traversal paths.

2.2.2. Knowledge Encoding. Let r_k be a rule for class c_k. Each rule is mapped using a single hidden node, modelling the conjunct, that connects the attributes corresponding to the appropriate pattern class. Therefore, one generates l hidden nodes for an l-class problem. The weight w_{ki}^1, between output node k and hidden node i, is set at $1 + \epsilon$, where ϵ is a small random number. The weight $w_{ia_j}^0$ between attribute a_j and hidden node i is clamped to $\frac{1}{Card(r_k)} + \epsilon$. Here $Card(r_k)$ indicates the number of features/attributes encountered along the traversal path from the root to the leaf of the decision tree containing the pattern corresponding to class c_k. In other words, $Card(r_k)$ is the number of operands in the conjunct of the rule r_k for class c_k.

3. Modular Knowledge-based Network

It is believed that the use of Modular Neural Network (MNN) enables a wider use of ANNs for large scale systems. Embedding modularity (i.e. to perform local and encapsulated computation) into neural networks leads to many advantages compared to the use of a single network. For instance, constraining the network connectivity increases its learning capacity, and permits its application to large scale problems [12]. Most logical rules are of the form **If . . . Then class k**, signifying the belongingness to a particular class. Each module/subnetwork takes care of one class, and can be represented by the relevant rule. Therefore, it is easier to encode *a priori* knowledge in modular neural networks. In addition, the number of network parameters can be reduced by using modularity. This feature speeds computation and can improve the generalisation capability of the system.

We use two phases. First, an l-class classification problem is split into l two-class problems. Let there be l sets of subnetworks, with n inputs and one output node each. Rough set theoretic concepts are used to encode domain knowledge into each of the subnetworks. The number of hidden nodes and connectivity of the knowledge-based subnetworks is automatically determined. A two-class problem leads to the generation of one or more crude subnetworks, each encoding a particular decision rule. Let each of these constitute a pool. So we obtain $m \geq l$ pools of knowledge-based modules. Each pool k is perturbed to generate a total of n_k subnetworks, such that $n_1 = \ldots = n_k = \ldots = n_m$. These pools constitute the initial population of subnetworks, which are then evolved independently using genetic algorithms.

At the end of training, the modules/subnetworks corresponding to each two-class problem are concatenated to form an initial network for the second phase. The inter-module links are initialised to small random values, as depicted in Fig. 1. A set of such concatenated networks forms the initial population of the GA. Note that the individual modules cooperate, rather than compete, with each other while evolving towards the final solution. The mutation probability for the inter-module links is now set to a high value, while that of intra-module links is set to a relatively lower value. This sort of *restricted* mutation helps preserve some of the localised rule

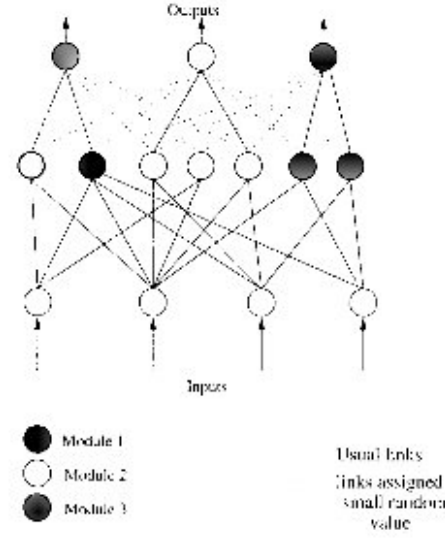

Fig. 1. Intra- and inter-module links.

structures, already extracted and evolved, as potential solutions. The initial population for the GA of the entire network is formed from all possible combinations of these individual network modules and random perturbations about them. This ensures that for complex multi-modal pattern distributions all the different representative points remain in the population. The algorithm then searches through the reduced space of possible network topologies.

4. Evolutionary Design

Genetic algorithms are highly parallel and adaptive search processes based on the principles of natural selection [15]. Here we use GAs for evolving the weight values, as well as the structure of the MLP used in the framework of modular neural networks. Unlike other theory refinement systems which train only the *best* network approximation obtained from the domain theories, the initial population here consists of all possible networks generated from rough set theoretic as well as ID3 rules. This is an advantage, because potentially valuable information may be wasted by discarding the contribution of less successful networks at the initial level itself.

Genetic algorithms involve three basic procedures – encoding of the problem parameters in the form of binary strings, application of genetic operators like crossover and mutation, selection of individuals based on some objective function to create a new population. Each of these aspects is discussed below with relevance to our algorithm. The block diagram depicting the interaction between the different hybrid components is provided in Fig. 2.

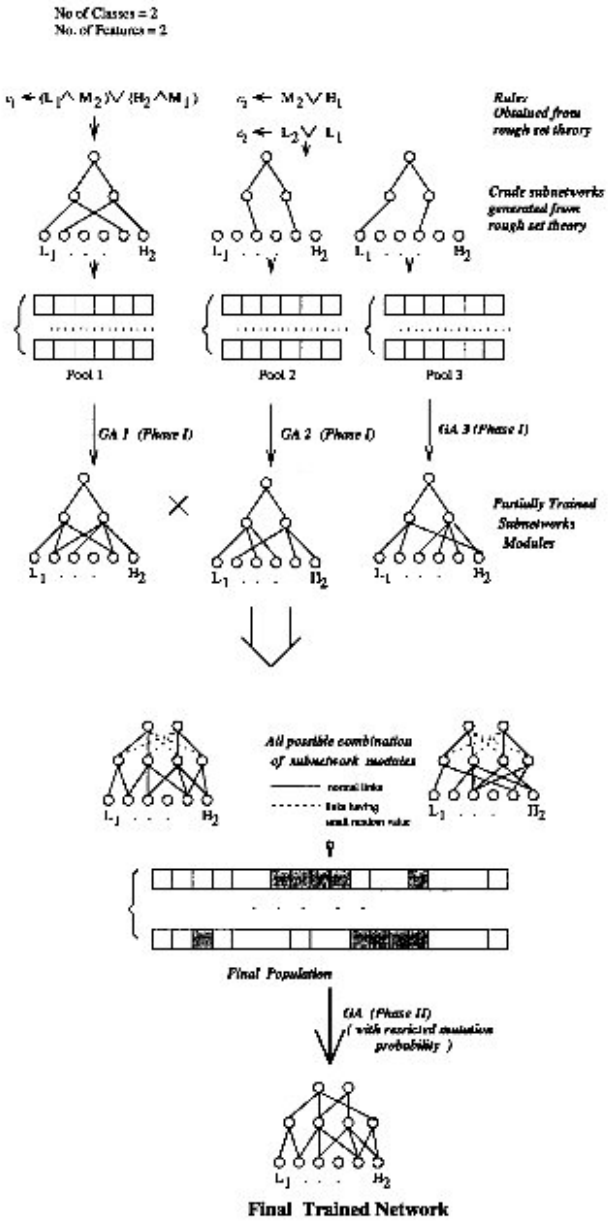

Fig. 2. Steps for designing a sample Modular Rough Fuzzy MLP.

4.1. Chromosomal Representation

The problem variables consist of the weight values and the input/output fuzzification parameters. Each of the weights is encoded into a binary word of 16 bits in length, where [000 . . . 0] decodes to −128 and [111 . . . 1] decodes to 128. An additional bit is assigned to each weight to indicate the presence or absence of the link. If this bit is 0, the remaining bits are unrepresented in the phenotype. The total number of bits in the string is therefore dynamic [17]. Thus, a total of 17 bits are assigned for each weight. Initial population is generated by coding the networks obtained by knowledge encoding, and by random perturbations about them. A population size of 64 was considered.

4.2. Genetic Operators

4.2.1. Crossover. It is obvious that, due to the large string length, single point crossover would have little effectiveness. Multiple point crossover is adopted to ensure a high probability for only one crossover point occurring within a word encoding a single weight. The crossover probability is fixed at 0.7.

4.2.2. Mutation. The search string being very large, the influence of mutation is more on the search. Each of the bits in the string is chosen to have some mutation probability (*pmut*). This

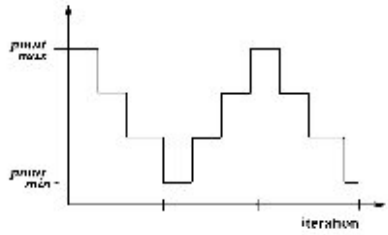

Fig. 3. Variation of mutation probability with iterations.

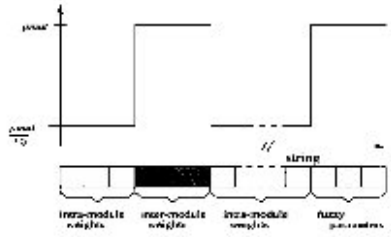

Fig. 4. Variation of mutation probability along the encoded string.

mutation probability, however, has a spatio-temporal variation. The variation of *pmut* with iterations is shown in Fig. 3. The maximum value of *pmut* is chosen to be 0.4 and the minimum value as 0.01. The mutation probabilities also vary along the encoded string as shown in Fig. 4, with the bits corresponding to inter-module links being assigned a probability *pmut* i.e. the value of *pmut* at that iteration) and intra-module links assigned a probability *pmut*/10. This is done to ensure least alterations in the structure of the individual modules already evolved.

4.3. Choice of Fitness Function

In GAs the fitness function is the final arbiter for string creation, and the nature of the solution obtained depends on the objective function. An objective function of the form described below is chosen:

$$F = \alpha_1 f_1 + \alpha_2 f_2 \tag{8}$$

where

$$f_1 = \frac{\textit{No. of Correctly Classified Sample in Training Set}}{\textit{Total No. of Samples in Training Set}}$$

$$f_2 = 1 - \frac{\textit{No. of links present}}{\textit{Total No. of links possible}}$$

Here α_1 and α_2 determine the relative weightage of each of the factors. α_1 is taken to be 0.9 and α_2 is taken as 0.1, to give more importance to the classification score compared to the network size in terms of number of links.

4.4. Selection

Selection is done by the *roulette wheel* method. The probabilities are calculated on the basis of ranking of the individuals in terms of the objective function, instead of the objective function itself. Fitness ranking overcomes two of the biggest problems inherited from traditional fitness scaling: *over compression* and *under expansion*. *Elitism* is incorporated in the selection process to prevent oscillation of the fitness function with generation. The fitness of the best individual of a new generation is compared with that of the current generation. If the latter has a higher value – the corresponding individual replaces a randomly selected individual in the new population.

5. Implementation and Results

The data consists of a set of 221 cervical cancer patient cases obtained from the database of the Chittaranjan National Cancer Institute (CNCI), Calcutta. Cross-validation of results is made with oncologists. There are four classes corresponding to Stages I, II, III and IV of the cancer, each containing 19, 41, 139, 19 patient cases, respectively. The input nodes/features of the proposed model represents the presence or absence of the symptoms, and the signs observed upon physical examination. The 21 boolean input features refer to *Vulva: healthy* (*Vu*(*h*)), *Vulva: lesioned* (*Vu*(*l*)), *Vagina: healthy* (*Va*(*h*)), *Vagina: spread to upper part* (*Va*(*u*)), *Vagina: spread middle part* (*Va*(*m*)), *Vagina: spread to lower part* (*Va*(*l*)), *Cervix: healthy* (*Cx*(*h*)), *Cervix: eroded* (*Cx*(*e*)), *Cervix: small ulcer* (*Cx*(*su*)), *Cervix: ulcerative growth* (*Cx*(*u*)), *Cervix: proliferative growth* (*Cx*(*p*)), *Cervix: ulcero -proliferative growth* (*Cx*(*l*)), *Paracervix: free* (*PCx*(*f*)), *Paracervix: infiltrated* (*PCx*(*i*)), *Urinary bladder base: soft* (*BB*(*s*)), *Urinary bladder base: hard* (*BB*(*h*)), *Rectrovaginal septum: free* (*RVS*(*f*)), *Rectrovaginal septum: infiltrated* (*RVS*(*i*)), *Parametrium: free* (*Para*(*f*)), *Parametrium: spread, but not upto* (*Para*(*nu*)) and *Parametrium: spread upto* (*Para*(*u*)), respectively.

Let the proposed methodology be termed Model S. The dependency rules used for knowledge encoding are obtained by two methods – using rough set theory and the ID3 algorithm. The performance of the methodologies is compared with that of an ordinary MLP (termed Model O), trained using backpropagation (BP) with weight decay.

Table 1. Crude rules obtained by rough set theory.

I	Cx(su) $\vee$ Para(f), Cx(p) $\vee$ Para(f), Cx(su) $\vee$ Para(nu)
II	Va(h) $\vee$ Cx(u), Va(h) $\vee$ Cx(l), Va(u) $\vee$ Cx(u), Para(nu), Pcx(f)
III	Para(nu), Para(u), Va(u) (Va(u) $\wedge$ Cx(u)) $\vee$ Cx(l) $\vee$ Va(m) (Va(h) $\wedge$ Cx(u)) $\vee$ (Va(u) $\wedge$ Cx(u)) $\vee$ Cx(l) (Va(u) $\wedge$ Cx(p)) $\vee$ Va(m) $\vee$ Cx(l)
IV	(Va(l) $\wedge$ Cx(u)) $\vee$ (Cx(u)) $\wedge$ Va(u)) $\vee$ (Va(l) $\wedge$ Para(u)) (Va(l) $\wedge$ Cx(p)) $\vee$ Va(m).

Table 2. Crude rules extracted via the ID3 algorithm.

I	Para(f) $\wedge$ Va(h) $\wedge$ Cx(u) Para(nu) $\wedge$ Va(h) $\wedge$ Cx(u) $\wedge$ PCx(f) $\wedge$ BB(s)
II	Va(u) $\wedge$ Para(f) $\wedge$ Cx(h) Va(h) $\wedge$ Pcx(i) $\wedge$ Cx(u) $\wedge$ BB(s) $\wedge$ Para(nu) Va(u) $\wedge$ Para(nu) $\wedge$ Cx(l) $\wedge$ BB(s)
III	Va(h) $\wedge$ Cx(l) $\wedge$ Para(u) Para(u) $\wedge$ Cx(u) $\wedge$ PCx(i) $\wedge$ BB(s) Va(u) $\wedge$ Cx(u) $\wedge$ Para(u)
IV	Va(l) $\wedge$ Cx(u) $\wedge$ Para(u) Va(m) $\wedge$ BB(h) $\wedge$ Cx(u) $\wedge$ Para(u) Va(m) $\wedge$ Cx(p) $\wedge$ BB(h) $\wedge$ Para(u).

5.1. Knowledge Encoding and Classification

The dependency rules generated via rough set theory and the ID3 algorithm, and used in the encoding scheme are provided in Tables 1 and 2. Recognition scores obtained for each of the data by the proposed modular network (Model S) are then presented and compared. Here 50% of the samples are used as training set, and the network is tested on the remaining samples.

These extracted rules are encoded to generate the knowledge-based MLP (Model S). Table 3 demonstrates the performance of Model S, using knowledge encoding by both rough sets and ID3 algorithm. It is observed to be superior, in both cases, to that of Model O. This can be corroborated from Fig. 5. Note that the results for Stage IV are poor due to the absence of sufficient training data.

Table 3. Comparative classification scores for different models.

Stage	Model O		Model S with knowledge encoding via			
			Rough set theory		ID3 algorithm	
	Train	Test	Train	Test	Train	Test
I(%)	65.00	64.70	65.00	64.70	90.00	89.71
II(%)	69.05	67.73	69.05	68.13	73.81	72.04
III(%)	93.66	93.01	94.13	90.02	90.14	90.02
IV(%)	42.11	40.09	44.21	41.87	42.11	40.19
Net(%)	80.97	79.23	81.02	79.52	82.74	80.02
# links	175		118		82	
Iterations	90		90		50	

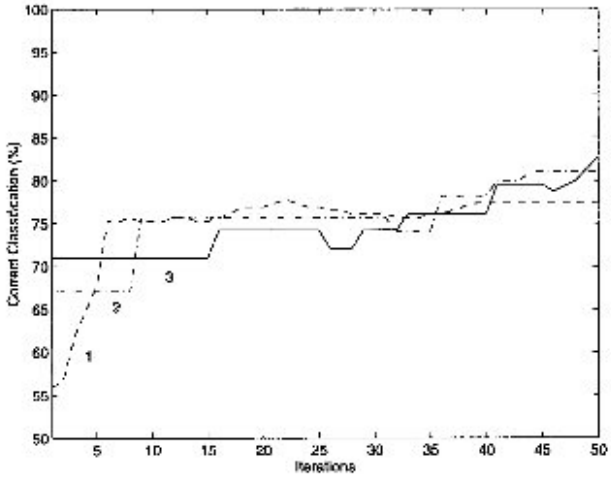

Fig. 5. Evolution of overall correct classification percentage with training sweeps. (1) Network trained with BP (Model O); (2) model S with initial rule encoding via rough set theory; (3) model S with initial rule encoding via ID3 algorithm.

5.2. Rule Extraction

Consider a simple heuristic for rule extraction. Let us define the following quantities: $Thres_1 = mean\ of\ the\ weights > 0$, $Thres_2 = mean\ of\ the\ weights > Thres_1$, $Thres_3 = mean\ of\ the\ weights > Thres_2$. We consider weights having value greater than $Thres_3$ as strong connections (plotted as thick lines in Fig. 6), weights having value between $Thres_2$ and $Thres_3$ as moderate links (plotted as normal lines in Fig. 6). We obtained $Thres_1 = 24.98$, $Thres_2 = 76.64$ and $Thres_3 = 85.09$. If the same set of threshold values are applied to Model O, no strong links are obtained. Hence, it is not possible to extract any crisp rules from it. On the other hand, the network obtained using the proposed Model S contains a number of strong links which can be used in extracting meaningful logical rules.

A sample set of refined rules extracted from the network, considering only the strong and moderate links, is presented below.

For a network with initial weight encoding from the crude rules obtained by rough set theory:

$$I \leftarrow (Va(h) \wedge \text{Para(f)}) \vee (\text{Cx(h)} \wedge \text{Cx(u)} \wedge \text{BB(s)})$$
$$II \leftarrow (PCx(f) \wedge \text{PCx(i)}) \vee \text{Para(f)} \vee \text{Para(nu)}$$
$$III \leftarrow Va(h) \wedge \text{Cx(u)} \wedge \text{Cx(l)} \wedge \text{Para(u)}$$

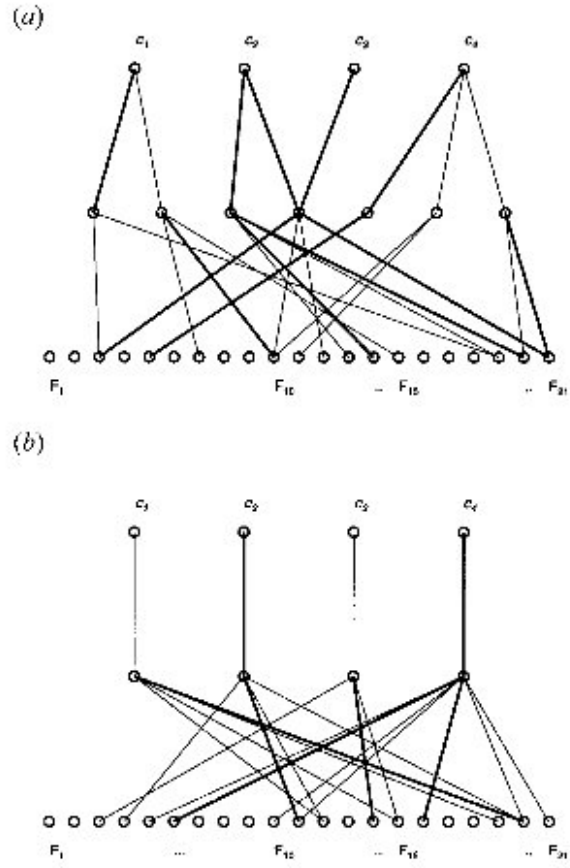

Fig. 6. Connectivity of the network obtained for the data, using Model S, with initial rule encoding via (a) rough set theory, and (b) ID3 algorithm.

$$IV \leftarrow Va(m) \vee (\mathrm{Cx(u)} \wedge \mathrm{Cx(p)}) \vee (\mathrm{Para(nu)} \wedge \mathrm{Para(u)}).$$

For a network with initial weight encoding from the crude rules obtained by the ID3 algorithm:

$$I \leftarrow Cx(u) \wedge \mathrm{Cx(l)} \wedge \mathrm{PCx(f)} \wedge \mathrm{BB(s)} \wedge \mathrm{Para(f)} \wedge \mathrm{Para(nu)}$$
$$II \leftarrow Va(h) \wedge \mathrm{Va(u)} \wedge \mathrm{Cx(p)} \wedge \mathrm{Cx(l)} \wedge \mathrm{Para(nu)} \wedge \mathrm{Para(u)}$$
$$III \leftarrow Va(h) \wedge \mathrm{Cx(u)} \wedge \mathrm{Cx(l)} \wedge \mathrm{PCx(i)} \wedge \mathrm{BB(s)}$$
$$IV \leftarrow Va(u) \wedge \mathrm{Va(m)} \wedge \mathrm{Va(l)} \wedge \mathrm{Cx(u)} \wedge \mathrm{Cx(p)} \wedge \mathrm{PCx(f)} \wedge \mathrm{BB(h)} \wedge \mathrm{Para(nu)} \wedge \mathrm{Para(u)}.$$

Here we provide the expertise obtained from oncologists. In Stage I the cancer has spread from the lining of the cervix into the deeper connective tissue of the cervix, but it is still confined within the cervix. Stage II signifies the spread of cancer beyond the cervix to nearby areas like parametrial tissue, that are still inside the pelvic area. In Stage III the cancer has spread to the lower part of the vagina or the pelvic wall. It may be blocking the ureters (tubes that carry urine from the kidneys to the bladder). Stage IV is the most advanced stage of cervical cancer. Now the cancer has spread to other parts of the body, like the rectum, bladder or lungs. It may be mentioned that the rules generated by the proposed algorithm are validated by the experts' opinion.

6. Conclusions

A methodology for detecting the different stages of cervical cancer using a modular knowledge-based network with genetic algorithms is presented. The proposed algorithm involves synthesis of several MLP modules, encoding rules generated by (1) rough set theory, and (2) ID3 algorithm, for a particular class. These knowledge-based modules are refined using a GA. The genetic operators are implemented in such a way that they help preserve the modular structure already evolved. It is seen that this methodology along with modular network decomposition results in superior performance in terms of classification score, training time and network sparseness, thereby enabling easier extraction of rules. The present investigation not only provides a decision support system for cervical cancer management, but also demonstrates a way how different *soft computing* tools like neural networks, GAs and rough set theory can be integrated to build an efficient decision making system for pattern classification and rule generation.

Acknowledgements. Authors thank Mr Pawan K. Singhal for his assistance in programming a part of the algorithm. This work was supported by CSIR Grant 25(0093)/97/EMR-II, India.

References

1. Brinton LA. Epidemiology of cervical cancer- overview, the epidemiology of cervical cancer and human papilloma virus. IACR 1992; 119: 4–23
2. Jayant K, Rao RS, Nene BM, Dale PS. Improved stage at diagnosis of cervical cancer with increased cancer awareness in a rural indian population. Int J Cancer 1995; 63: 161–163
3. National Cancer Registry Programme. Biennial Report, ICMR, India, 1988–89
4. Fu LM. Knowledge-based connectionism for revising domain theories. IEEE Trans Systems, Man and Cybernetics 1993; 23: 173–182
5. Towell GG, Shavlik JW. Knowledge-based artificial neural networks. Artificial Intelligence 1994; 70: 119–165
6. Ivanova I, Kubat M. Initialisation of neural networks by means of decision trees. Knowledge-Based Systems 1995; 8: 333–344
7. Pawlak Z. Rough Sets, Theoretical Aspects of Reasoning about Data. Kluwer Academic, 1991
8. Slowiński R. (ed.) Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory. Kluwer Academic, 1992

9. Pal SK, Skowron A. (eds.) Rough Fuzzy Hybridisation: New Trends in Decision Making. Springer-Verlag, Singapore, 1999
10. Pao YH. Adaptive Pattern Recognition and Neural Networks. Addison-Wesley, Reading, MA, 1989
11. Quinlan JR. C4.5, Programs for Machine Learning. Morgan Kaufman, 1993
12. Happel BM, Murre JJ. Design and evolution of modular neural network architectures. Neural Networks 1994; 7: 985–1004
13. Murray-Smith R. A Local Model Network approach to non-linear modelling. PhD, thesis, Department of Computer Science, University of Strathclyde, UK, 1994
14. Hansen L, Salamon P. Neural network ensembles. IEEE Trans Pattern Analysis and Machine Intelligence 1990; 12: 993–1001
15. Goldberg DE. Genetic Algorithms in Search, Optimisation and Machine Learning. Addison-Wesley, Reading, MA, 1989
16. Pal SK, Wang PP. (eds.) Genetic Algorithms for Pattern Recognition. CRC Press, Boca Raton, 1996
17. Maniezzo V. Genetic evolution of the topology and weight distribution of neural networks. IEEE Trans Neural Networks 1994; 5: 39–53
18. Yao X, Liu Y. A new evolutionary system for evolving artificial neural networks. IEEE Trans Neural Networks 1997; 8(3): 694–713
19. Andrews R, Diederich J, Tickle AB. A survey and critique of techniques for extracting rules from trained artificial neural networks. Knowledge-Based Systems 1995; 8: 373–389
20. Banerjee M, Mitra S, Pal SK. Rough fuzzy MLP: Knowledge encoding and classification. IEEE Trans Neural Networks 1998; 9(6): 1203–1216