Available online at www.sciencedirect.com
SCIENCE DIRECT®
Fuzzy Sets and Systems 150 (2005) 351–371
FUZZY sets and systems
www.elsevier.com/locate/fss

Genetically optimized logic models

Witold Pedrycz[a, b, *], Marek Reformat[a]

[a] *Department of Electrical & Computer Engineering, University of Alberta, 238, Civil/Electrical Engineering Building, Edmonton, Canada T6G 2G7*
[b] *Systems Research Institute, Polish Academy of Sciences, 01-447 Warsaw, Poland*

Received 3 August 2002; received in revised form 30 December 2003; accepted 18 May 2004

Abstract

This study is concerned with the genetic development of logic-based fuzzy models (logic networks). The models are constructed with the aid of AND and OR fuzzy neurons. These neurons are logic-driven processing units realizing s–t and t–s composition operations of fuzzy sets and fuzzy connections with "t" and "s" being triangular norms and co-norms. The fundamental architecture of the network comprises a hidden layer formed by a collection of AND neurons that is followed by an output layer consisting of OR neurons. We show that this structure of the network directly translates into a collection of "if-then" statements. The learning procedure in such networks consists of two main development phases. First, a blueprint of the network is constructed genetically (through the use of a genetic algorithm) with the underlying intent of capturing the structural (logical) essence of the data and encapsulating it into the network. At this development phase the resulting network is kept binary (Boolean) with anticipation that the binary connections are a sound model of the blueprint of the logic nature of the data. The second phase of the design of the network is aimed at it refinement in which we move from the binary connections and optimize them to reflect the details of the data. There are two main optimization mechanisms applied in the above sequence: the structural binary development uses a genetic algorithm (GA) while a parametric refinement of the network is realized through a gradient-based learning. This organization of the optimization process exhibits several evident advantages. The GA concentrates on a preliminary and rough binary optimization and in this way helps combat an eventual curse of dimensionality. This curse is inherent to high-dimensional problems to which gradient-based learning is quite vulnerable. Having the blueprint developed, further refinement being guided by gradient-based computing becomes more effective. An interesting optimization aspect addressed in the genetic optimization environment deals with a construction of optimal subspaces of the overall feature space: in highly dimensional spaces we can envision that only a small subset of features is essential and their reduction may help develop a compact network. We introduce a concept of network connectivity to quantify the aspect of feature reduction. A carefully presented numerical example serves as a detailed illustration of the development approach.

* Corresponding author. Department of Electrical and Computer Engineering, University of Alberta, 238, Civil/Electrical Engineering Building, Edmonton, Canada T6G 2G7. Tel.: +1-7804923332; fax: +1-7804921811.
E-mail address: pedrycz@ee.ualberta.ca (W. Pedrycz).

0165-0114/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.fss.2004.05.009

Keywords: Logic-based models; Genetic optimization; Gradient-based learning; Two-phase development strategy; Structural blueprint; Boolean structure; Logic processor; Feature selection; Model transparency; Network connectivity; Computational Intelligence (CI)

1. Introduction

The ongoing research challenges in system modeling lie in the transparency and accuracy of models. These two main design requirements are highly conflicting. In the realm of fuzzy modeling, there have been a number of main pursuits that attempted to meet these two fundamental requirements. Interestingly, a way in which they are satisfied depends very much upon a synergy realized between fuzzy sets and neural networks—two fundamental contributors to the area of neurofuzzy models. The leading role of neural networks shifts emphasis on the learning abilities, higher approximation and accuracy with lower interpretability of the resulting models. When fuzzy sets become dominant, the interpretability and descriptive aspects of the model increase that happens at some expense of accuracy. Striking a suitable balance (where the concept of suitability is profoundly application driven) is a delicate matter and requires the right mix of technologies of fuzzy sets and neural networks. The panoply of existing fuzzy (or neurofuzzy) and genetic models are an evident testimony to this diversity of synergies going on in this area, cf. [2,3,5,6,8–10,14,15,17]. The addition of the genetic optimization to fuzzy modeling is important and its role in global optimization is an evident asset of the Computational Intelligence (CI) framework formed in this way.

The logic-based fabric of fuzzy models is an inherent facet of this modeling paradigm that need to be preserved. While neurocomputing is an important technology contributing to the learning abilities and subsequently to the parametric optimization of the fuzzy models, its is of paramount importance to develop a synergistic environment in which the logic transparency of the ensuing models is not compromised. The idea behind fuzzy neurons and fuzzy neural networks introduced and studied in [11] was to develop a structure that is at the same time transparent and adaptive. The transparency facet is gained due to the logic type of processing realized by the neurons. There are two fundamental classes of the neurons that is OR and AND neurons. Interestingly, these are generalizations of standard OR and AND gates encountered in digital systems. The neurons exhibit learning abilities as they come with a collection of adjustable connectives (weights). In this setting of fuzzy neurons, the synergy of learning and transparency is well articulated. Different application-oriented aspects of the resulting fuzzy neural networks were discussed in detail in a number of previous studies [11,12].

An important design issue these networks face deals with the learning efficiency in the networks. In particular, we are interested in investigating the behavior of such networks in case of highly variable models. An impact of problem dimensionality on fuzzy models in general (including those of rule-based structure) is still an open matter (as it becomes quite apparent from the literature, fuzzy models are quite “modest” in terms of the number of the variables handled). The objective of this study is to introduce a comprehensive learning environment. We anticipate that with the increasing number of variables, some standard gradient-based learning applied to the fuzzy neural networks will start loosing efficiency quite quickly when the dimensionality of the problem start increasing. In a nutshell, we need to resort ourselves to more comprehensive learning schemes. Evolutionary computing is a legitimate avenue to explore here. The global nature of the genetically inclined search and its potential scalability are important features worth considering with the regard to the problem.

In this study, we propose a unified genetic development environment for fuzzy neural networks. The underlying methodology comprises two important phases. In the first phase, we are after a binary two-valued blueprint of the network. Here the role of genetic optimization is profound. More specifically, we consider genetic algorithms (GAs) to be used to construct the blueprint of the network. The second phase is about a refinement of the binary structure. As the skeleton of the network has been already formed, the gradient-based learning becomes more efficient even for large architectures as the connections present in the binary version are refined and adjusted by following the gradient of the performance index.

In the numeric experiments we use a Boston dataset available at the Machine Learning (UC at Irvine, http:// www.ics.uci.edu/∼mlearn/MLSummary.html). In this data, we treat a price of real estate as an output variable while considering all remaining variables to be the inputs. In this form of the model we predict the price on a basis of a number of characteristics of real estate such as year, distance from main employment centers, pollution, etc. The example will be developed across the entire study and discussed in subsequent sections. In this way we can fully illustrate the concepts and specific architectural developments of the model as they become introduced.

The material is arranged into seven sections and its organization follows the main phases of the genetic development of the models. The topology of fuzzy models with a clear specification of their main functional modules is discussed in Section 2. This is followed by a detailed analysis and design of logic-oriented processing implemented by means of fuzzy neurons (Section 3). In particular, we highlight a number of essential learning challenges that are associated with the dimensionality of the problem (number of variables) and severely affect the gradient-based learning in the logic networks. A two-phase design of the logic networks is studied in Section 5; here we analyze how genetic optimization leads to a blueprint (Boolean skeleton) of the network and how this structure becomes afterwards optimized (refined) by gradient-based learning mechanisms. Conclusions are covered in Section 6.

The terminology and notation used within this study are standard. We use capital letters to denote fuzzy sets. The logic operators on fuzzy sets are realized with the aid of t- and s-norms. In all numeric experiments we consider these triangular norms and co-norms to be a product operation and probabilistic sum, respectively.

2. The topology of fuzzy models

In general, fuzzy models comprise of two fundamental functional components: (a) input and output interfaces and (b) a processing core. Fuzzy models being in essence a *conceptual* and logic-driven construct interact with a *physical* world of measurable variables. The input and output interface are used to make this interaction between the two worlds possible. The input interface realizes a perception process: we perceive and transform input variables (their specific numeric or granular values) into an internal format of information granules being a language of the entities to be processed at the logic level. The output interface communicates the results of the logic processing to the external world of numeric data. Quite commonly following a long tradition in fuzzy controllers we refer to these interfaces as a fuzzification (input interface) and defuzzification (output interface) (Fig. 1).

There have been a number of theoretical and practical investigations into the nature of the interfaces where numerous issues concerning the number of fuzzy sets, their distribution and ensuing optimization have been discussed, cf. [11,12]. Quite visible was a discussion on setting up meaningful selection criteria guiding processes of defuzzification. Apparently, the optimization of the fuzzy sets standing in these

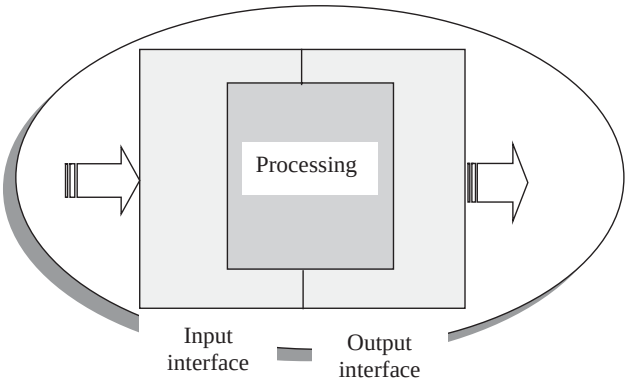

Fig. 1. A general topology of fuzzy models with clearly identified functional modules of input interfacing, logic processing at the level of information granules and output interfacing with the external world. Note a visible distinction between the worlds of physical and conceptual entities (fuzzy sets).

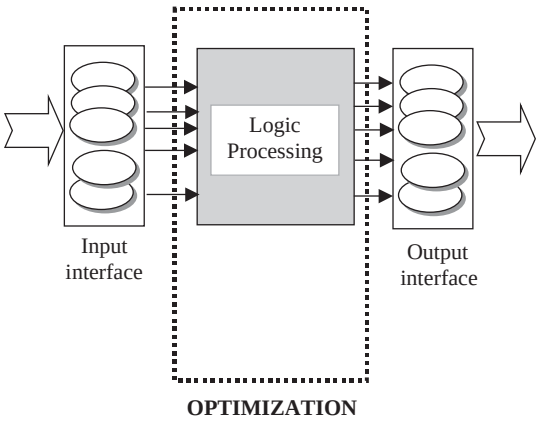

Fig. 2. The optimization mode of the fuzzy model realized at the internal level of the processing core operating on membership grades.

interfaces could be beneficial to the model and contribute to the enhancement of its quality. Accepting this point of view, for the purpose of this study we assume that fuzzy sets of the input and output interfaces are given in advance and left unchanged. Our focal point is the logic processing core of the fuzzy model and its optimization. One can portray this identification scheme as shown in Fig. 2; note that we assume that the experimental data become available to the processing part after their transformation through the interfaces. This simply means that the optimization of the processing core is guided by some performance available at this internal level of the model as shown in Fig. 2 (which means that we carry out optimization at the level of the membership grades).

We do not claim that the selection and number of fuzzy sets forming the interfaces are not important in general. For the purpose of this study, we keep this part fixed and this allows us to concentrate on the core processing realized in fuzzy models.

Moving on with the fuzzy model of the Boston housing data, all variables are transformed by means of Gaussian like membership function described as $\exp(-(x-m)^2/\sigma^2)$ with "m" standing for the modal value of the linguistic term (fuzzy set) and σ being its spread. The experiments are completed for 3 and 4 fuzzy sets defined for each variable being uniformly distributed across the corresponding universes of discourse. The spread is chosen in such a way that two successive fuzzy sets overlap. The degree of overlapping is controlled via the level at which two neighboring membership functions intersect. In the experiments presented in this study this overlap is set at the level of 0.5.

3. The logic-based core of fuzzy models: a design with fuzzy neurons

The ultimate objective of a fuzzy model is to convert data into a logically transparent topology of the model. To meet this objective, we have to confine ourselves to generic components of the model that exhibit an evident logic character of processing. This is accomplished by building a model with the use of logic-based neurons. They seamlessly combine the logic type of processing and learning abilities residing supported by the adjustable parameters of these neurons.

3.1. Logic processing units- AND and OR fuzzy neurons

The AND and OR fuzzy neurons were introduced in [11] as basic logic processing elements. The basic formulas governed the functioning of these elements are constructed with the aid of t- and s-norms. AND neuron

$$y = \mathop{\mathrm{T}}_{i=1}^{n} (w_i \mathrm{s} x_i) \tag{1}$$

OR neuron

$$y = \mathop{\mathrm{S}}_{i=1}^{n} (w_i \mathrm{t} x_i), \tag{2}$$

where $x_1, x_2, \ldots$ and x_n are inputs while $w_1, w_2, \ldots, w_n$ denote connections of the neurons. Both the inputs and connections (weights) are confined to the unit hypercube. Note that (1)–(2) are just t–s and s–t compositions (convolutions) of two discrete fuzzy sets. We also use a shorthand notation by collecting the inputs and connections in two vectors ($\mathbf{x}$ and $\mathbf{w}$, respectively). This leads to the expression $y = \mathrm{AND}(\mathbf{x}; \mathbf{w})$ and $y = \mathrm{OR}(\mathbf{x}; \mathbf{w})$. Some obvious observations hold:

- for binary inputs and connections, the neurons transform to standard AND and OR gates
- the higher the values of the connections in the OR neuron, the more essential the corresponding inputs. This observation helps eliminate irrelevant inputs: the inputs associated with the connections whose values are below a certain threshold are eliminated. An opposite relationship holds for the AND neuron: here the connections close to zero identify the relevant inputs.
- The change in the values of the connections of the neuron is essential to the development of the learning capabilities of a network formed by such neurons; this parametric flexibility is an important feature to be exploited in the design of the networks.

These two types of fuzzy neurons are fundamental building blocks used in the design of logic expressions supporting the development of logic-driven models. The weights equip the logic neurons

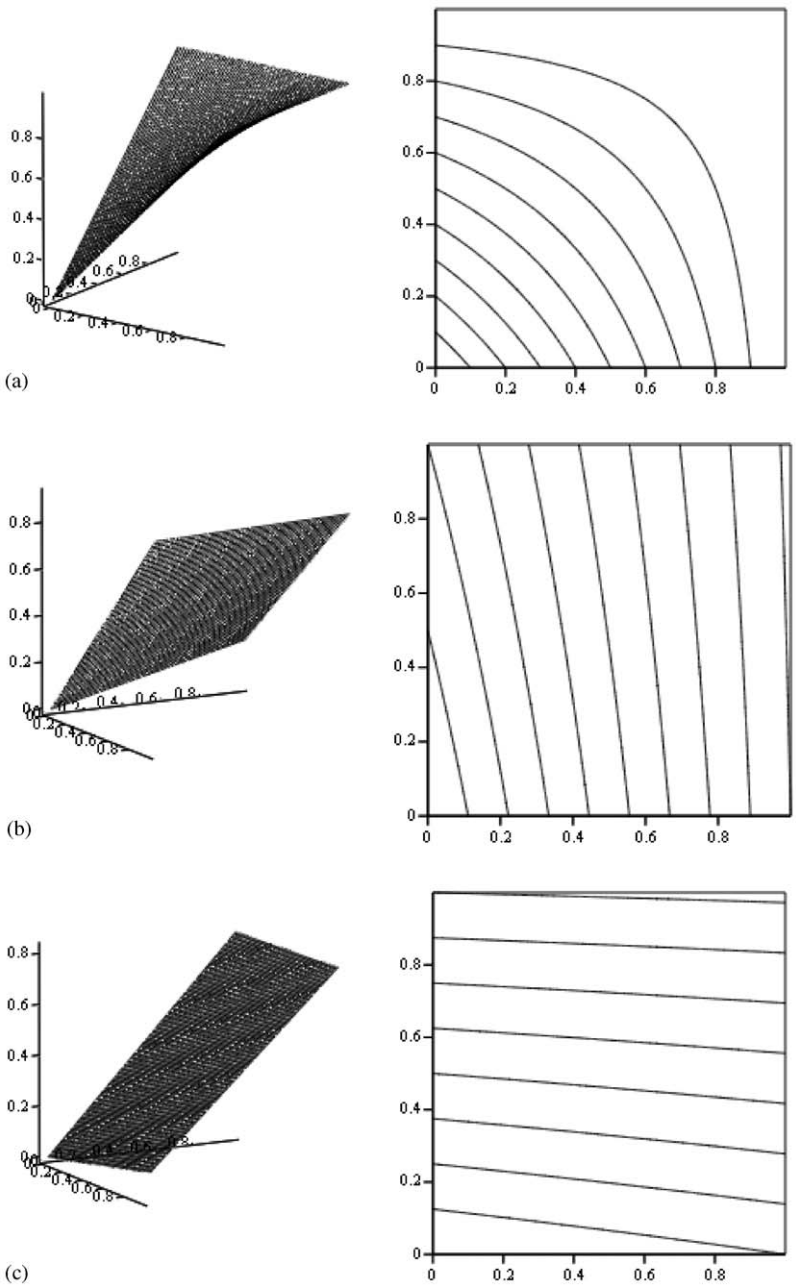

Fig. 3. Characteristics of the OR neurons for selected combinations of the connections: (a) $\mathbf{w} = [1\ 1]$ (b) $\mathbf{w} = [0.9\ 0.2]$, (c) $\mathbf{w} = [0.1\ 0.8]$.

with a significant parametric flexibility; as shown in Figs. 3 and 4 the changes in the connections affect their characteristics. This becomes quite profound when inspecting their 2D plots of input–output relationships.

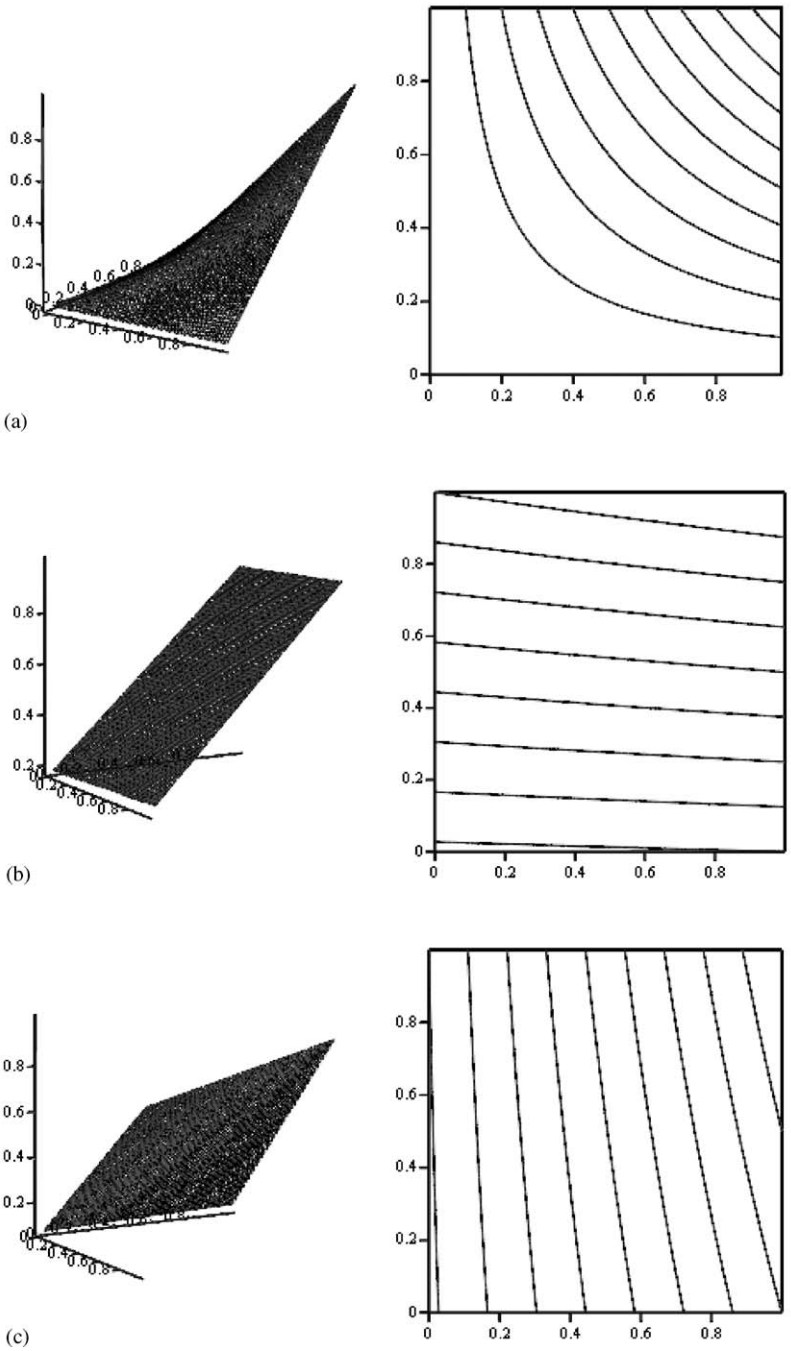

Fig. 4. Characteristics of the AND neurons for selected combinations of the connections: (a) $\mathbf{w} = [0\ 0]$, (b) $\mathbf{w} = [0.9\ 0.2]$, (c) $\mathbf{w} = [0.1\ 0.8]$.

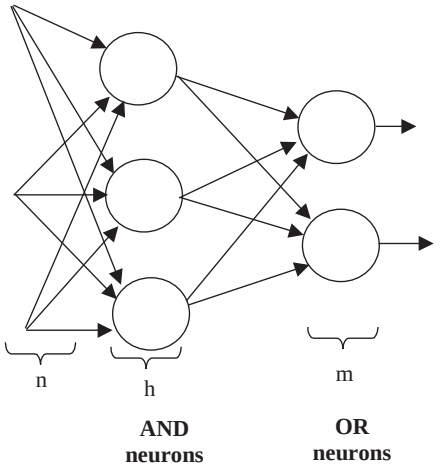

Fig. 5. A general architecture of the logic-based network; refer to the text for the detailed notation.

3.2. The topology of the logic-based processing core of the fuzzy model

The evident transparency of the model is realized by adhering to the logic fabric of its building blocks. In a nutshell, we use logic neurons being viewed here as a generic means of forming the skeleton (blueprint) of the logic model. The topology of the model comes as a realization of the logic expressions capturing the experimental data. In a nutshell, the rule-based description of data is a collection of if-then statements combined altogether or-wise. The rules map directly onto the two-layer structure of the network, Fig. 5. The first layer consisting of AND neurons form a collection of conditions of the rules. The OR neurons in the consecutive layer are aimed at the aggregation of the rules having the same conclusion.

The network is fully described by two matrices of connections (**W** and **V**) with the first one describing the AND neurons while the second one captures the OR neurons. The above matrices collect the connections of the individual neurons so we treat $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_h$ and $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m$ as vectors of the connections of the individual neurons. It means that the ith AND neuron reads as $z_i = \text{AND}(\mathbf{x}; \mathbf{w}_i)$. Similarly, the notation $y_j = \text{OR}(\mathbf{z}; \mathbf{v}_j)$ applies to the jth neuron in the output layer ($j = 1, 2, \ldots, m$).

4. Learning in fuzzy models: challenges and limitations

The gradient-based learning of this class of the fuzzy model comes with solid experimental evidence as to the nature of the optimization. More importantly, the experimental studies help us quantify the limits of this type of learning.

4.1. Computational challenges and dimensionality of the problem

The dimensionality of the problem arises as a major obstacle to the efficient learning of the network. It is of interest to get a better and quantitative insight into the problem and understand why the learning mechanisms collapse when the dimensionality of the problem increases. The critical part of the network

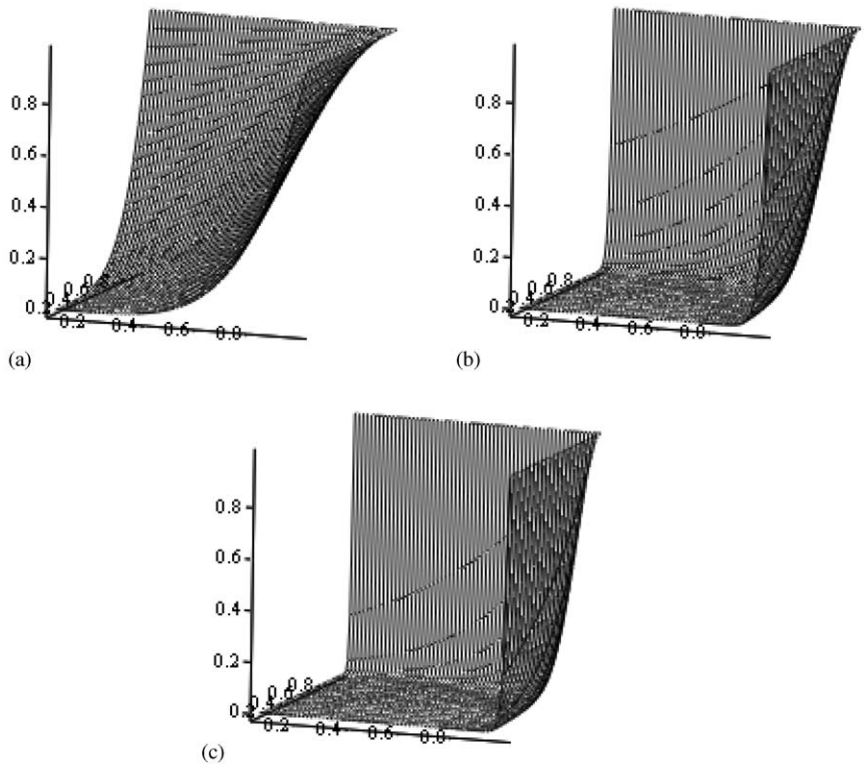

Fig. 6. Plots of the output of the AND neuron treated as a function of w and x for selected values of n: $n = 5$ (a), $n = 50$ (b), and $n = 100$ (c).

are the AND neurons in the hidden layer. The number of the inputs of these neurons is directly implied by the number of the input variables or being more precise their representations in terms of the fuzzy sets defined for each original variable, refer to Fig. 2. To quantify the dimensionality effect, we express the output of the AND neuron treated as a function of the number of its inputs (n). Similarly, we express the derivative of the output of this neuron taken with respect to the connection. To simplify the analysis, let us consider all connections being the same (and equal to w). The same assumption is made with regard to the inputs (these are viewed equal to x). Furthermore we specify t-norm as a product operation and take the s-norm to be a probabilistic sum. Under these assumptions, the output of the neuron (z) reads as

$$z = (x \text{ s } w)^n$$

while its derivative is expressed as follows:

$$\frac{\mathrm{d}z}{\mathrm{d}w} = (x \text{ s } w)^{n-1}(1 - x).$$

The plots of these two relationships for selected number of inputs are shown in Figs. 6 and 7. Noticeably, both z and its derivative are heavily affected by the increasing values of "n"; at some point both of them turn out to be practically equal to zero. The character of these relationships explains why the learning becomes inherently inefficient if not impossible for the higher dimensionality of the problem.

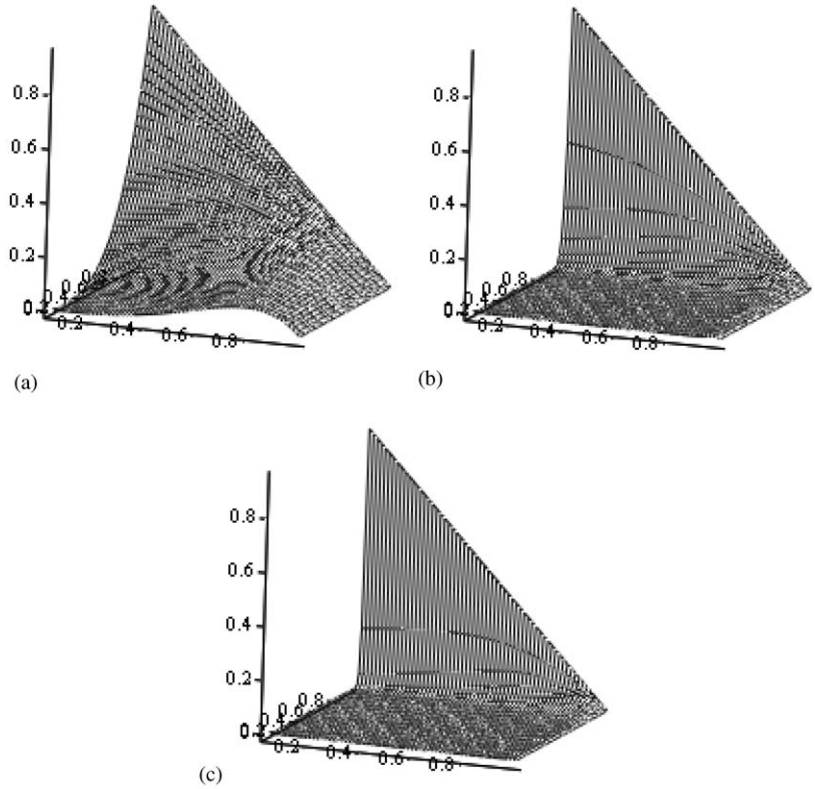

Fig. 7. Plots of the derivative of the AND neuron taken with respect to the connection and treated as a function of w and x for selected values of n: $n = 5$ (a), $n = 50$ (b), and $n = 100$ (c).

To illustrate this point, we consider a part of the Boston data treating it as a learning set and proceed with a standard gradient-based learning. The learning is monitored by recording the values of the performance index (sum of squared errors). The fuzzy discretization (quantization) involves three fuzzy sets for each variable. This results in $3^*12 = 36$ input variables and 3 output variables. In the experiments, the size of the hidden layer is equal to 5 and 6 with an intent to investigate an effect this parameter of the structure has on the efficiency of the learning process. The learning rate is equal to 0.05 (we keep this value low on purpose; the experience shows that for higher values, the learning may not converge). As becomes apparent, see Fig. 8, the learning exhibits an interesting pattern: there is no significant optimization for a number of initial iterations, then occurs a rapid jump in the performance index and after this the performance does not improve. The learning requires more iterations (learning epochs) with the increased size of the hidden layer.

With the increase of the size of the hidden layer, there is no convergence. We view this as a result of the size of the network. A similar lack of convergence was observed for higher numbers of inputs to the network. For instance, with four fuzzy sets defined for each variable (that leads to 48 input variables), the learning was not successful (no convergence was reported).

To alleviate this problem, we have to develop a different learning strategy in which the gradient-based learning is augmented by some structural learning.

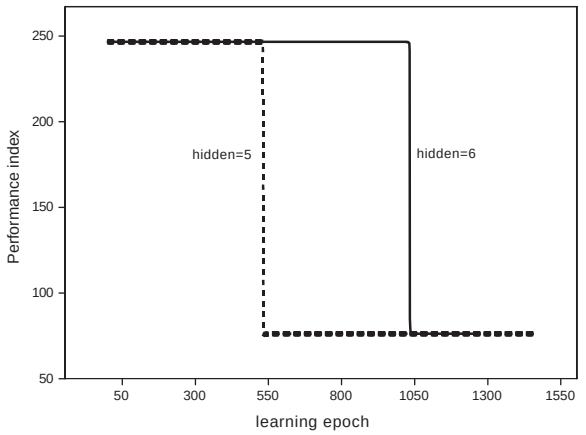

Fig. 8. Learning in the fuzzy neural network for two selected sizes of the hidden layer (5 and 6 AND neurons); $a = 0.05$ and three fuzzy sets for each variable.

5. A two-phase design of the logic fuzzy models

Having identified the shortcomings of the learning in the fuzzy model, it becomes obvious that further pursuits require a far more comprehensive optimization policy. What comes as an intuitively appealing approach is a hybrid learning methodology comprising of two fundamental techniques, namely genetic optimization and gradient-based optimization.

5.1. Structure and skeleton optimization via a genetic algorithm

To battle the dimensionality problem, we attempt to construct a skeleton of the network by detecting the most essential connections that shape up the architecture and then concentrate on the detailed optimization of these connections. As the blueprint of the network has to capture the essence of the logic mapping, for the purpose of this optimization, the connections are binary (0-1). The optimization has to be global. In light of these two requirements, genetic optimization is a suitable avenue to follow. More specifically, we consider a standard version of a genetic algorithm (GA) [1,5,7,15]. The binary string is a representation of the structure of the network. As the connections are two-valued, the entries of the string map directly on the connections of the neurons. The arrangement of the connections is also evident as visualized in Fig. 9. The treatment of the connections as binary entities is highly desirable. By doing this, we concentrate our effort on the determination of the most essential structure of the model that attempts to reveal the crux of its topology. We anticipate that such structure will hold in spite of small variations in data and therefore could be regarded as a Boolean "skeleton" of the processing core of the model.

Bearing in mind the form of AND and OR processing realized by the logic neurons, it becomes obvious that the binary connections of the neuron have a straightforward interpretation, refer also to Fig. 5. In case of the AND neuron: If the connection is equal to zero, this means that the corresponding input impacts the output of this neuron. For the OR neuron a reverse situation occurs: the connection equal to one implies that the specific input is essential and affects the output of the neuron. Noticeably, under these binary conditions these two neurons behave like digital AND and OR gates.

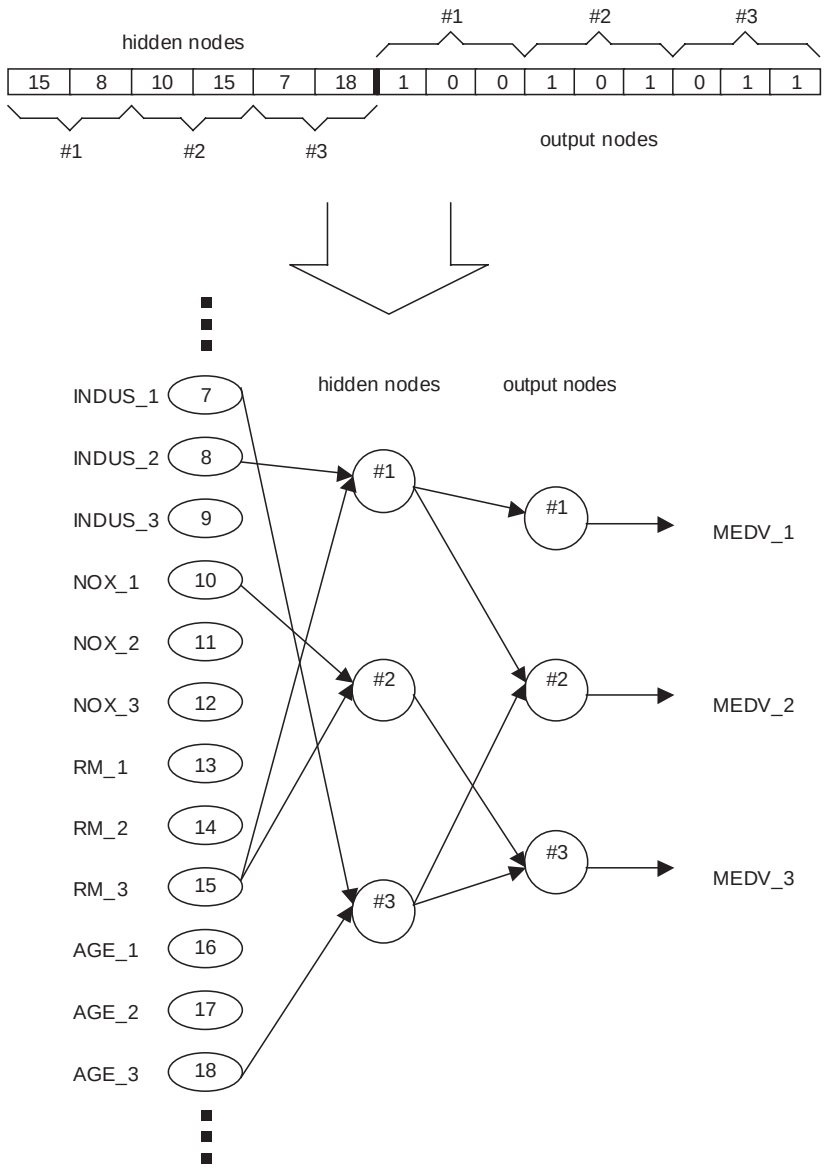

Fig. 9. A part of the network and its one-to-one correspondence with the binary string to be optimized by GA. The details of the structure of the network pertain to the Boston housing data with 36 inputs (12 input variables with 3 fuzzy sets per variable) 3 outputs (that is 3 fuzzy sets defined for the output variable) and connectivity of 2 inputs per node in the hidden layer (suffixes _1, _2 and _3 represent linguistic labels *small*, *medium* and *large*, respectively).

From the design standpoint, we are at position to control the development of the topology of the network. In particular, we can fix the level of connectivity between the input and hidden layer. We may refer to a maximal number of connections from the input nodes to each node (AND neuron) in the hidden layer as the network *connectivity*. For instance, if we have two inputs to the first AND neuron, four to the second and three to the third, we say that the connectivity of the neuron is equal four.

Table 1
Parameters of GA used in the experiment

Parameter	Value
Population size	100
Selection process	tournament
Mutation rate	0.8
Crossover rate	0.3

Such connectivity analysis helps understand the nature of data. First, the number of the meaningful (zero) connections of the neuron determines a local dimensionality of the input space. The contributing inputs form this *local* space. The dimensionality of such space can vary from neuron to neuron. By varying the maximal number of the connections that are allowed, we can gain a better insight into the intrinsic dimensionality of the data space. The connections of the OR neuron indicate which of these local input spaces contribute to the output of the network.

An initial configuration of the Boolean skeleton of the network is set up as follows. For the given number of the neurons in the hidden layer and the connectivity level (that is the number of the inputs going to each neuron in the hidden layer), we randomly initiate the connections of the AND neurons (making sure that the number of the connections between the inputs and each neuron is as specified). Initially, the connections of the OR neurons are all equal to one. Then we allow the genetic optimization to adjust the connections.

Proceeding with the numeric data, we carry the genetic optimization of the network. The parameters of the GA used in the experiment are summarized in Table 1 (note that these parameters were chosen experimentally). The fitness function maximized by the GA is expressed in the form of the sum of squared errors between target values (data) and outputs of the network. This fitness has to be minimized (it stands in contrast with the "classic" way of using fitness functions in GAs whose values are maximized; if necessary a simple transformation of taking a negative value of the fitness, -fitness, brings us in line with the way of maximizing the fitness).

We carried out a series of experiments for two parameters controlling the structure of the network, that is a level of connectivity and the number of AND nodes in the hidden layer. The results are presented in Table 2 and these concern both the training and testing set. The performance index reported there is taken as an average error per data point. This measure helps us compare the results independently from the size of the training and testing data sets.

We note that for a fixed connectivity, the increase of the size of the hidden layer results (in general) in lower values of the performance index (as we can learn by scanning the columns of the two tables). It is interesting to learn that the increased connectivity for a fixed size of the hidden layer gives rise to the higher values of the performance index (and this tendency is very evident across all rows of the tables). These two observations hold true for the increased number of the fuzzy sets, refer to Table 3.

The way in which the genetic optimization led to the optimization of the model is reported through the values of the fitness function in successive generations (Figs. 10 and 11) (note that the fitness function is minimized as we have already emphasized in the earlier discussion).

As already indicated, the "meaningful" connections of the AND neurons are those equal to zero. For the OR neurons, the connections equal to 1 are essential. The architectural investigations lead us to two interesting and important design observations

Table 2
The results of the GA optimization (performance index) of the network for the training (a) and testing (b) set (3 membership functions, case in boldface are used in further optimization of the model)

Hidden\connectivity	2	3	4	5
(a) *training data set—after GA, before tuning*				
2	0.0784	0.0850	0.0909	0.1267
3	0.0604	0.0644	0.0853	0.1097
4	0.0590	0.0661	0.0816	0.1131
5	0.0617	0.0615	0.0842	0.1121
6	**0.0577**	0.0660	0.0829	0.1081
7	0.0584	0.0607	0.0818	0.1064
8	0.0587	0.0622	0.0684	0.1087
9	0.0586	0.0614	0.0735	0.1024
10	0.0591	0.0623	0.0737	0.1000
(b) *testing data set—after GA, before tuning*				
2	0.0812	0.0868	0.0906	0.1269
3	0.0646	0.0649	0.0869	0.1064
4	0.0648	0.0690	0.0853	0.1055
5	0.0655	0.0648	0.0835	0.1127
6	**0.0625**	0.0673	0.0874	0.1046
7	0.0655	0.0644	0.0792	0.1048
8	0.0591	0.0638	0.0726	0.1066
9	0.0601	0.0653	0.0750	0.1011
10	0.0637	0.0650	0.0759	0.1002

Table 3
The results of the GA optimization of the network for the training (a) and testing (b) set (4 membership functions per variable)

Hidden\input	2	3	4	5
(a) *training data set—after GA, before tuning*				
2	0.0764	0.1023	0.1134	0.1403
3	0.0708	0.0882	0.1144	0.1494
4	0.0655	0.0851	0.1102	0.1386
5	0.0724	0.0898	0.1099	0.1339
6	**0.0628**	0.0767	0.1111	0.1413
7	0.0724	0.0817	0.1067	0.1406
8	0.0654	0.0858	0.1104	0.1333
9	0.0669	0.0786	0.1098	0.1358
10	0.0742	0.0848	0.1184	0.1351
(b) *testing data set—after GA, before tuning*				
2	0.0675	0.0945	0.1084	0.1393
3	0.0650	0.0836	0.1099	0.1554
4	0.0542	0.0759	0.1047	0.1361
5	0.0667	0.0869	0.1073	0.1289
6	**0.0552**	0.0648	0.1077	0.1411
7	0.0666	0.0710	0.1028	0.1389
8	0.0546	0.0833	0.1053	0.1278
9	0.0619	0.0740	0.1057	0.1304
10	0.0649	0.0792	0.1150	0.1359

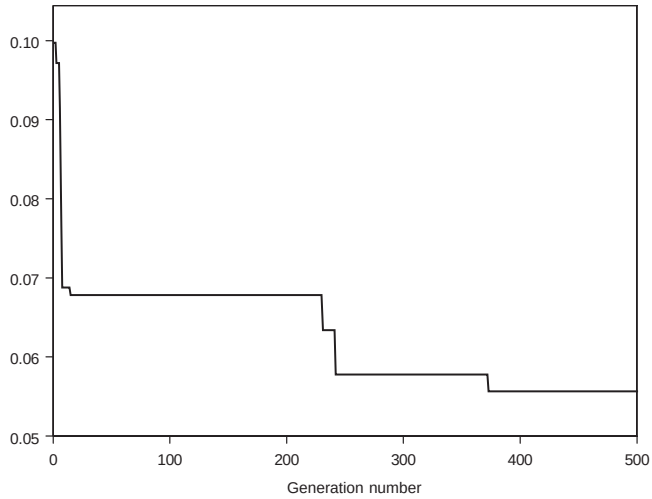

Fig. 10. Performance index–fitness function (expressed as error per data point) for training data set in successive generations (3 membership functions, the final value for training data set is 0.056 that is lower than the result for the testing data set being equal to 0.066).

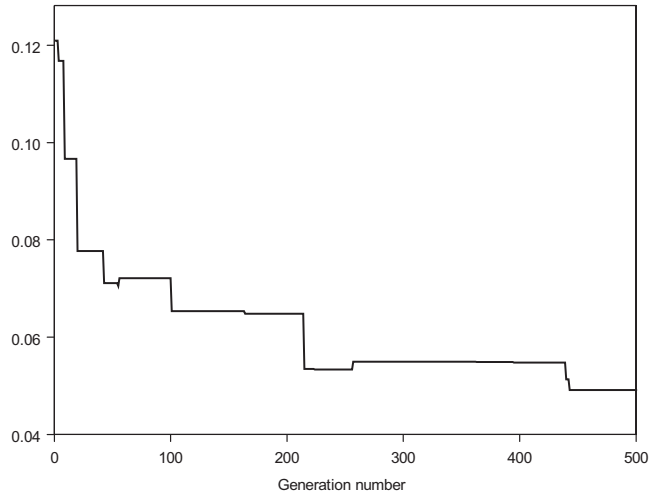

Fig. 11. Performance index–fitness function (expressed as error per data point) for training data set in successive generations (4 membership functions, final value for training data set 0.060, the result for the testing data set is equal to 0.049).

AND neurons form local feature spaces: The zero connections of these neurons identify the inputs that are essential locally for the given neuron. Such local feature space whose dimensionality is usually far lower than the original space is termed in this way as it pertains to the region being captured by the AND neuron. The local feature spaces help us rank features (more specifically fuzzy sets being the inputs to the processing core of the model) as to their importance to the overall model. A simple count of the occurrence of the features in the local spaces coming with the AND neurons is sufficient in this regard.

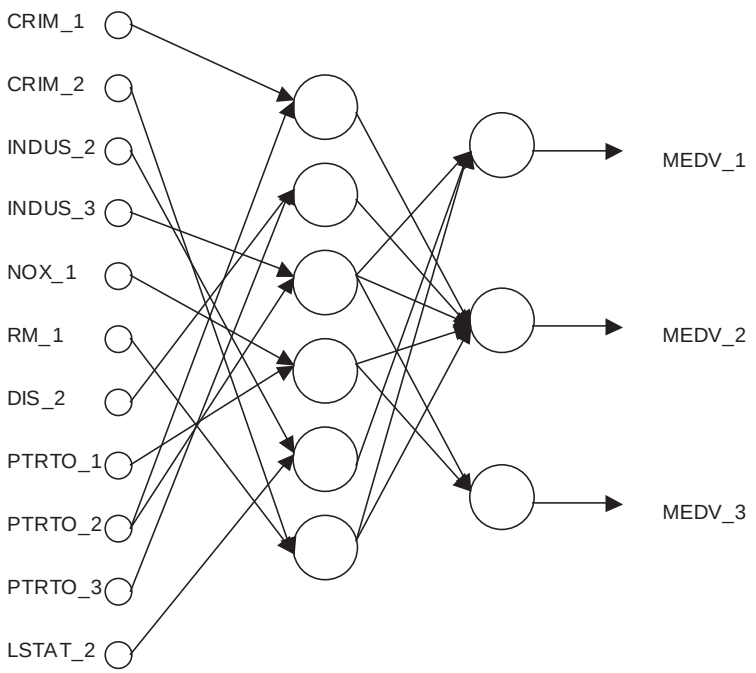

Fig. 12. A network generated by GA optimization process (with 3 membership functions per variable).

As each AND neuron may have a different collection of the inputs, the input space is quite heterogeneous by changing its properties from neuron to neuron.

OR neurons aggregate local feature spaces: The features originated at the level of the AND neurons are combined by the OR neuron. Some subspaces are discarded if the corresponding connection of the neuron is set to zero by the genetic optimization.

The Boolean blueprint of the network translates immediately into a series of rules (Fig. 12). For instance taking a rule with conclusion MEDV_1, we read the following rule:

if
NON-RETAIL BUSINESS is *large* (INDUS_3)
and
PUPIL-TEACHER RATIO is *medium* (PTRTO_2)
or
NON-RETAIL BUSINESS is *medium* (INDUS_2)
and
% OF LOWER STATUS POPULATION is *medium* (LSTAT_2)
or
CRIME RATE is *medium* (CRIM_2)
and
AVERAGE ROOM NUMBER is *small* (RM_1)
then
MEDIAN VALUE OF HOME is *small* (MEDV_1)

In the sequel, we discuss the second development phase of the fuzzy model.

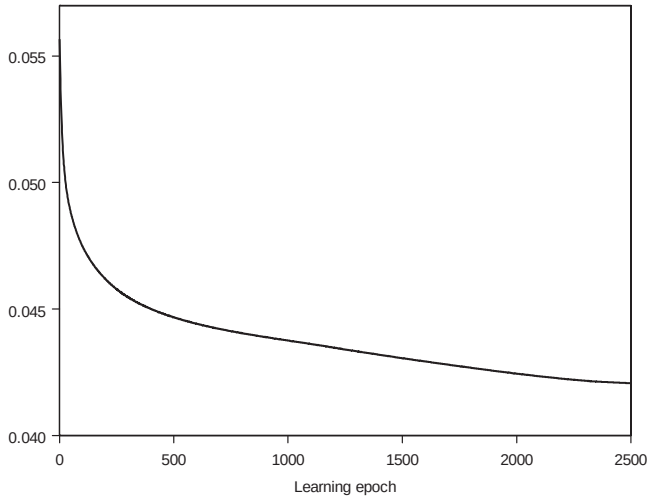

Fig. 13. Performance index for training data set in successive learning epochs (3 membership functions, final value for training data set is equal to 0.042, the value of the performance index for the testing data is the same as for the training data).

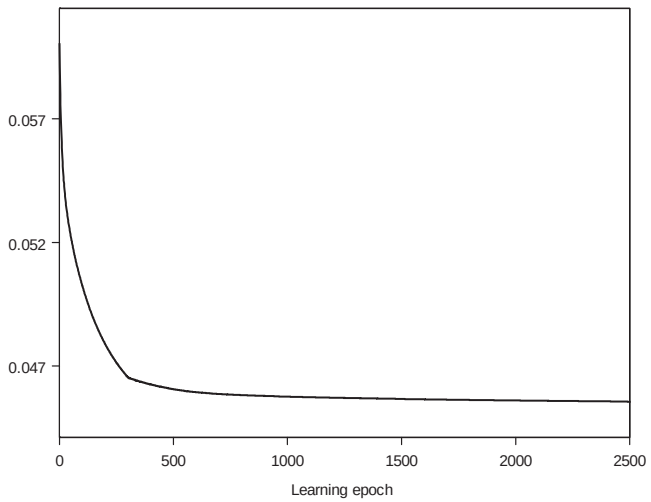

Fig. 14. Performance index for training data set in successive learning epochs (4 membership functions, training data — 0.046 testing data — 0.041).

5.2. Gradient-based optimization as parametric refinement of the logic structure

The GA produces a Boolean blueprint of the network. In this way some initial values of the connections have been established. The data have been approximated by a Boolean formula. The role of the gradient-based learning is to take from this point and move on with fuzzy set refinement. The refinement involves transforming Boolean connections into the weights in the unit interval. This enhancement aims at further reduction in the value of the performance index (Figs. 13, 14).

Table 4
The results of the parametric optimization of the network for the training (a) and testing (b) set (3 membership functions, case in boldface will be used in further discussion)

Hidden\input	2	3	4	5
(a) *training data set—after tuning*				
2	0.0540	0.0522	0.0466	0.0435
3	0.0403	0.0435	0.0454	0.0473
4	0.0438	0.0412	0.0404	0.0434
5	0.0412	0.0401	0.0429	0.0408
6	**0.0438**	0.0424	0.0381	0.0434
7	0.0392	0.0343	0.0373	0.0410
8	0.0380	0.0348	0.0402	0.0386
9	0.0406	0.0419	0.0369	0.0406
10	0.0353	0.0350	0.0358	0.0326
(b) *testing data set—after tuning*				
2	0.0548	0.0520	0.0474	0.0467
3	0.0412	0.0451	0.0456	0.0478
4	0.0459	0.0428	0.0403	0.0451
5	0.0429	0.0407	0.0426	0.0409
6	**0.0439**	0.0429	0.0368	0.0442
7	0.0376	0.0307	0.0363	0.0415
8	0.0372	0.0317	0.0412	0.0377
9	0.0372	0.0429	0.0356	0.0410
10	0.0331	0.0322	0.0361	0.0309

To move from the Boolean structure obtained through genetic optimization to the gradient-based learning, we introduce a mask **M**. The mask involves two matrices whose dimensionality coincides with the size of the network as we started with. Its role is to focus on learning the connections that have been identified through the genetic optimization as candidates for further optimization. The gradient-based learning can be expressed in the concise manner as follows:

$$\textbf{connection} = (\textbf{connection})_{\mathbf{M}} - \alpha(\nabla_{\textbf{connection}} \mathrm{Q})_{\mathbf{M}},$$

where α is a positive learning rate while $(.)_{\mathbf{M}}$ underlines that the connections identified by the mask are affected. The mask appearing in the above expression states that the connections that have been identified as meaningful during the genetic optimization are further refined while the remaining are not optimized whatsoever. The minimized performance index Q is a standard sum of squared errors between the output of the network and the target (training data). In the above expression, we use the shorthand notation **connection** to collect all connections of the network. More specifically, more detailed formulas distinguish between the OR and AND neurons and the connections to be modified

- for AND neurons the mask points at the connections close to zero that may be further refined with the update of the (i, j)th connection ΔW_{ij} expressed in the form with M_{ij} being the corresponding entry of the mask **M**

$$\Delta W_{ij} = \begin{cases} 0 & \text{if } M_{ij} = 0, \\ \frac{\partial \mathrm{Q}}{\partial W_{ij}} & \text{if } M_{ij} \neq 0, \end{cases}$$

Table 5
The results of the parametric optimization of the network for the training (a) and testing (b) set (4 membership functions, case in boldface to be used in further discussion)

Hidden\input	2	3	4	5
(a) *training data set—after tuning*				
2	0.0650	0.0645	0.0653	0.0583
3	0.0512	0.0483	0.0586	0.0457
4	0.0473	0.0474	0.0501	0.0553
5	0.0530	0.0519	0.0531	0.0486
6	**0.0454**	0.0453	0.0491	0.0481
7	0.0447	0.0455	0.0419	0.0447
8	0.0426	0.0402	0.0493	0.0462
9	0.0433	0.0421	0.0426	0.0448
10	0.0417	0.0422	0.0466	0.0419
(b) *testing data set—after tuning*				
2	0.0570	0.0553	0.0593	0.0500
3	0.0472	0.0450	0.0515	0.0409
4	0.0428	0.0434	0.0458	0.0500
5	0.0469	0.0469	0.0491	0.0410
6	**0.0418**	0.0410	0.0434	0.0459
7	0.0405	0.0388	0.0368	0.0410
8	0.0376	0.0373	0.0471	0.0417
9	0.0422	0.0393	0.0406	0.0390
10	0.0375	0.0371	0.0428	0.0396

Table 6
The weights of the GA-generated network, Fig. 12, after tuning, only for connections between inputs and nodes from hidden layer (*x* represents no connections, gray entries indicate valid connections—with weights lower than 0.5)

	In #1 CRIM_1	In #2 CRIM_2	In #3 INDUS_2	In #4 INDUS_3	In #5 NOX_1	In #6 RM_1	In #7 DIS_2	In #8 PTRTO_1	In #9 PTRTO_2	In #10 PTRTO_3	In #11 LSTAT_2
Hidden node #1	0.00	x	x	x	x	x	x	x	0.39	x	x
Hidden node #2	x	x	x	x	x	x	0.00	x	x	0.40	x
Hidden node #3	x	x	x	0.00	x	x	x	x	0.25	x	x
Hidden node #4	x	x	x	x	0.43	x	x	0.14	x	x	x
Hidden node #5	x	x	0.64	x	x	x	x	x	x	x	0.00
Hidden node #6	x	0.00	x	x	x	0.72	x	x	x	x	x

- for the OR neurons the mask identifies the connections that are usually close to one and are still subject to optimization

$$\Delta V_{ij} = \begin{cases} 0 & \text{if } M_{ij} = 0, \\ \frac{\partial Q}{\partial V_{ij}} & \text{if } M_{ij} \neq 0. \end{cases}$$

For the detailed learning formulas the reader may refer to [12]. The intensity of the overall refinement of the network can be expressed by computing the following difference:

$$\|\mathbf{W}[g] - \mathbf{W}^{\sim}\| + \|\mathbf{V}[g] - \mathbf{V}^{\sim}\|,$$

Table 7
The weights of the GA-generated network, Fig. 12, after tuning, only for connections between nodes from hidden layer and outputs ("*x*" represents no connection, gray entries indicate valid connections whose values are higher than 0.5)

	Hidden node #1	Hidden node #2	Hidden node #3	Hidden node #4	Hidden node #5	Hidden node #6
Out #1 MEDV_1	x	x	0.67	x	0.52	1.00
Out #2 MEDV_2	0.78	0.77	0.05	0.31	x	0.00
Out #3 MEDV_3	x	x	0.00	0.59	x	x

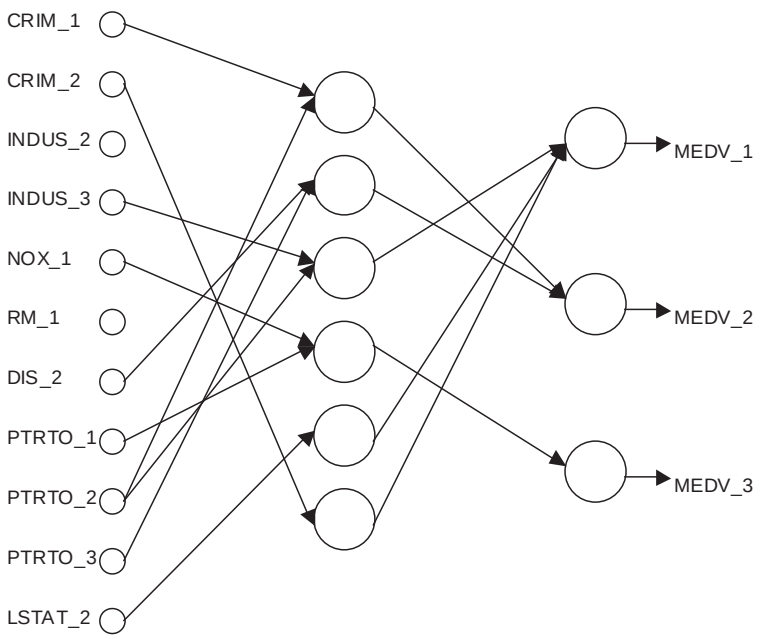

Fig. 15. A network generated by GA optimization process and tuned by parametric optimization (case for 3 membership functions). Shown are only "valid" connections, viz. those indicated by the gray entries from the two previous tables.

where $\|.\|$ denotes a distance between two matrices of the connections. The above expression states how much the network was refined by computing a total departure from the Boolean connections. More specifically we distinguish between the connections that have been genetically optimized ($\mathbf{W}[g]$ and $\mathbf{V}[g]$ that are binary) and those arising after the gradient-based refinement, that is $\mathbf{W}^{\sim}$ and $\mathbf{V}^{\sim}$. The higher the values of this index, the more modifications to the connections were necessary to enhance the quality of the network during the gradient-based optimization phase (Tables 4–7).

By contrasting Fig. 15 with the Boolean structure we note that the refinement of the structure has resulted in several connections being eliminated (or equivalently rules being reduced).

6. Conclusions

The proposed development of fuzzy models is an example of essential synergy of the contributing technologies of Computational Intelligence. The architecture seamlessly combining the logic fabric of

fuzzy set operations with parametric flexibility of the logic neurons is optimized in a genetic environment. We have a comprehensive design environment consisting of two main phases of genetic (global) and gradient-based optimization (parametric refinement). The important concept of network connectivity associated with local properties of the input space has been introduced and used as an essential mechanism of dimensionality reduction.

There are several open issues worth pursuing that have not been directly tackled in this study. First, the matter of the input and output interfaces has been left out from the study. One can anticipate that the careful design of the interfaces, see [13], can further enhance the performance of the fuzzy model. The related question is that of simultaneous or separate optimization of the input and output interfaces. The ongoing challenge of dealing with fuzzy models of high dimensionality is another point worth investigating. Further extensions along the line of evolutionary optimization such as evolutionary programming and evolutionary strategies are potential optimization vehicles to be exploited in the future studies.

Acknowledgements

Support from the Natural Sciences and Engineering Research Council (NSERC) and Alberta Software Engineering Research Consortium (ASERC) is gratefully acknowledged.

References

[1] T. Back, D.B. Fogel, Z. Michalewicz (Eds.), Evolutionary Computations I, Institute of Physics Publishing, Bristol, 2000.
[2] A. Bastian, Identifying fuzzy models utilizing genetic programming, Fuzzy Sets and Systems 113 (3) (2000) 333–350.
[3] S. Bengio, Y. Bengio, J. Cloutier, Use of genetic programming for the search of a learning rule for neural networks, in: Proc. of the First Conf. on Evolutionary Computation, IEEE World Congr. on the Computational Intelligence,1994, pp. 324–327.
[4] A.I. Esparcia-Alcázar, K.C. Sharman, Evolving recurrent neural network architectures by genetic programming, Genetic Programming 97, in: Proc. of the Second Ann. Conf., Stanford Univ., USA, July 1997, pp. 89–94.
[5] D.B. Fogel, Evolutionary Computation, Toward a New Philosophy of Machine Intelligence, IEEE Press, Piscataway, 1995.
[6] T. Furuhashi, S. Matsushita, H. Tsutsui, Evolutionary fuzzy modeling using fuzzy neural networks and genetic algorithm, in: Proc. of 1997 IEEE Internat. Conf. on Evolutionary Computation (ICEC'97), 1997, pp. 623–627.
[7] J.H. Holland, H. John, Adaptation in Natural and Artificial Systems, 2nd Edition, MIT Press, Cambridge, MA, 1992.
[8] C.L. Karr, E.J. Gentry, Fuzzy control of pH using genetic algorithms, IEEE Trans. Fuzzy Systems 1 (2) (1993) 46–53.
[9] D. Kim, C. Kim, Forecasting time series with genetic fuzzy predictor ensemble, IEEE Trans. Fuzzy Systems 5 (5) (1997) 523–535.
[10] P.G. Korning, Training neural networks by means of genetic algorithms working on very long chromosomes, Internat. J. Neural Systems 6 (3) (1995) 299–316.
[11] W. Pedrycz, Fuzzy Sets Engineering, CRC Press, Boca Raton, FL, 1995.
[12] W. Pedrycz, Computational Intelligence: An Introduction, CRC Press, Boca Raton, FL, 1997.
[13] W. Pedrycz, Fuzzy equalization in the construction of fuzzy sets, Fuzzy Sets and Systems 119 (2) (2001) 329–335.
[14] W. Pedrycz, M. Reformat, Rule-based modelling of nonlinear relationships, IEEE Trans. Fuzzy Systems 5 (2) (1997) 256–269.
[15] M. Russo, FuGeNSys: a genetic neural system for fuzzy modeling, IEEE Trans. Fuzzy Systems 6 (3) (1998) 373–388.
[16] M. Russo, Genetic fuzzy learning, IEEE Trans. Evol. Comput. 4 (3) (2000) 259–273.
[17] X. Yao, Evolving artificial neural networks, Proc. IEEE 87 (9) (1999) 1423–1447.