Available online at www.sciencedirect.com
SCIENCE DIRECT®
Engineering Applications of Artificial Intelligence 17 (2004) 383–391
Engineering Applications of
ARTIFICIAL
INTELLIGENCE
www.elsevier.com/locate/engappai

Logic-driven fuzzy modeling with fuzzy multiplexers

Witold Pedrycz*

Department of Electrical & Computer Engineering, University of Alberta, Edmonton, Canada T6G 2G7
and Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland

Abstract

We introduce a concept of a fuzzy multiplexer, discuss its use in the design of logic networks and elaborate on the role of such networks in fuzzy modeling. In essence, a fuzzy multiplexer, fMUX, acts as a fuzzy switch whose output is determined on a basis of the logic values of the information inputs being switched (selected) by the select input. Multiplexers are generic building modules in digital systems. We show that networks of fuzzy multiplexers can play a similar role in fuzzy modeling. The two general design methodologies are studied and contrasted. The first is based on gradient-based learning and helps carry out parametric optimization. The second approach exploits a global genetic optimization and supports a structural and parametric development of the network. This becomes especially attractive in case of multivariable systems where the number of system variables has to be reduced. Experimental studies are reported for synthetic Boolean data and continuous (multivalued) problems.

Keywords: Fuzzy multiplexers; Genetic optimization; Digital systems; Feature selection; Fuzzy modeling; Multivalued logic

1. Introduction: from digital systems to fuzzy logic in system design

One can state without any exaggeration that the fundamentals of fuzzy modeling are inherently rooted in the world of multivalued or fuzzy logic. The underlying logic nature of the models makes them transparent and user-centric. The same transparency contributes to a highly interpretative insight into experimental data. The agenda of fuzzy modeling is inherently associated with the transparency of fuzzy models. While this facet of modeling has already started to gain visibility and properly balance the otherwise accuracy-driven fuzzy models, there are still a number of fundamental issues as to the definition of interpretability itself, granularity of models vis-à-vis the characteristics of experimental data, and assessment of the readability of the structure of the model itself (Bargiela and Pedrycz, 2002; Delgado et al., 1997; Gomez-Skarmeta et al., 1999; Zadeh and Kacprzyk, 1999).

Two-valued logic forms a well-known boundary case of the fuzzy logic. The design of digital systems comes with a diversity of well-established, highly efficient and scalable architectures and related development algorithms. By acknowledging a point of view that the two-valued logic is just a special case of fuzzy logic, we are then tempted to generalize or reformulate the already existing architectures and design practices of digital systems and cast them in the framework of fuzzy logic. This point of view is the crux of this study. We concentrate on a standard technique of implementation of combinational systems by means of multiplexers (the approach which results in an array of multiplexers implementing any Boolean function) and generalize this concept to the world of fuzzy logic. In essence, we are concerned with the three main phases: (a) building a generic structure of a fuzzy multiplexers, (b) developing models (networks) exploiting fuzzy multiplexers as their building components (we will be referring to them as networks of fuzzy multiplexers), and (c) designing such networks with the aid of methods of structural and parametric optimization.

As emphasized, the embedding principle (where fuzzy logic subsumes two-value logic and inherits from its fundamental constructs) makes this starting point of

*Corresponding author. Department of Electrical & Computer Engineering, University of Alberta, Edmonton, Canada T6G 2G7 Tel.: +1-204-474-8380; fax: +1-204-261-4639.
E-mail address: pedrycz@ee.ualberta.ca (W. Pedrycz).

0952-1976/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2004.04.011

view especially justifiable and appealing considering that multiplexers have been commonly used in the design of digital systems and come with a well-developed design methodology (Ciletti, 1999; Kohavi, 1970; Mano, 1991; McCluskey, 1986).

The organization of the material is structured in a way it reflects the research agenda outlined above. First, we introduce a basic processing module of a fuzzy multiplexer and discuss its characteristics (Section 2). This naturally leads us to the networks formed by fMUXs; Section 3 relates their structure to the expansion theorem by Shannon and emphasizes the nature of function decomposition completed in this manner. Section 4 is concerned with the general development of fMUX networks and serves as a prerequisite to the comprehensive discussion on the design of the networks. Section 5 concentrates on the genetic optimization of the networks that helps address an issue of structural optimization (concerned with a selection of an optimal subset of input variables) and their parametric learning. The discussion covers all architectural considerations of genetic algorithms (GAs) and presents the underlying genetic operators pertinent to the optimization realized here. Experimental results are shown in Section 6.

The terminology used here adheres to the standards used in two-valued logic, digital systems, and fuzzy logic. The logic operators are modeled via t- and s-norms. If not stated otherwise, in this study we use two standard realizations of t- and s-norms in the form of a product and probabilistic sum. The motivation behind their selection is twofold. In contrast to the commonly used minimum and maximum operators these give rise to smooth input–output relationships. Secondly, they relate to some probabilistic constructs in this manner could link to the operations being used in the probability calculus (intersection and union of random events). An overbar symbol denotes a complement treated in a usual way encountered in logic (that is $\bar{x} = 1\text{-}x$).

2. Fuzzy multiplexer as a generic processing unit

We are concerned with the development of logic-based models of data in the unit hypercube. More precisely, such models realize logic transformations that map the unit hypercubes (say $[0,1]^n$ into $[0,1]$) and come with some well-defined semantics. We require that such mapping is made modular, meaning that it is built on the basis of a collection of simple processing units (nodes). The basic processing node, referred to as a fuzzy multiplexer, fMUX, realizes a mapping from $[0,1]^2$ into $[0,1]$ and is governed by the expression

$$y = c_0 * \bar{x} + c_1 * x, \tag{1}$$

where the logic operations ($*$ and $+$) are implemented using some t- and s-norms. In other words, (1) is implemented as

$$y = (c_0 t \bar{x}) s (c_1 t x). \tag{2}$$

Schematically, the structure of (1) can be illustrated as shown in Fig. 1. The variable (x) standing in the above expression plays the role of a *switching* (selection or select) variable that allows two fixed *information* inputs (c_0 or c_1) that affect the output. The degree to which the produced result depends on these fixed information values is controlled by the select variable. To emphasize the role played by all these signals, we use a concise notation $y = \text{fMUX}(x, \mathbf{c})$, where $\mathbf{c}$ ($=[c_0\ c_1]$) denotes a vector of the information inputs. In the two boundary conditions the select variable may assume, we produce a binary switch (the same as being used in digital systems). It means that if $x = 1$, then $y = c_1$. Likewise, the value of x set to 0 leads to the output being equal to c_0 meaning that the value of c_0, is transferred to the output of the device.

Fig. 2 includes a series of plots of the characteristics of the fuzzy multiplexer being treated as a function of x; noticeable is a fact that different configurations of the values of the information inputs (c_0 and c_1) give rise to different nonlinear input–output relationships of the device. By choosing a certain value of the select input, we logically "blend" the two constant logic values present at the information inputs.

Using fuzzy multiplexers, we can easily form cascade structures (networks) as commonly encountered in the two-valued logic constructs (digital systems). As an

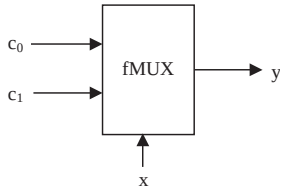

Fig. 1. A schematic representation of the fuzzy multiplexer with one select input (x) and two fixed information inputs (c_0 and c_1).

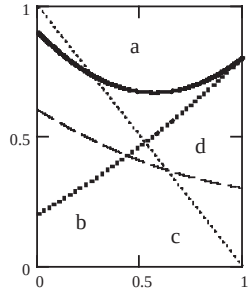

Fig. 2. Input–output characteristics of the fuzzy multiplexer for selected values of c_0 and c_1, $y = \text{fMUX}(x,c_1,c_0)$: *a* - $c_0 = 0.9$ $c_1 = 0.8$; *b*- $c_0 = 0.2$, $c_1 = 0.8$; *c*–$c_0 = 1.0$, $c_1 = 0.0$; *d*- $c_0 = 0.6$, $c_1 = 0.3$. In all cases t-norm is treated as a product and s-norm is implemented as the probabilistic sum.

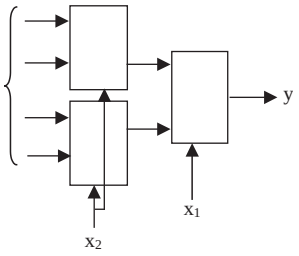

Fig. 3. A two-layer network of fuzzy multiplexers.

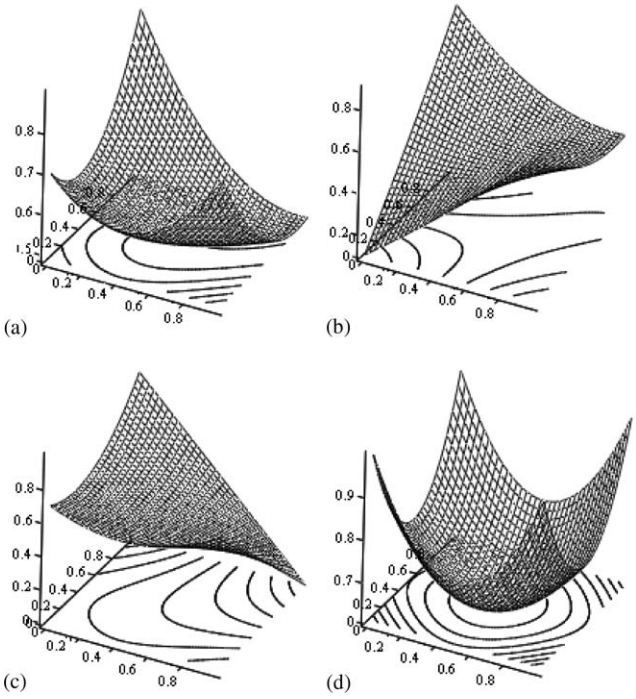

Fig. 4. 3D plots of the characteristics of the fMUX for selected combinations of the parameters and t- and s-norms realized as a product and probabilistic sum: $\mathbf{c}=[0.7\ 0.8\ 0.9\ 0.5]$ (a); $\mathbf{c}=[0.1\ 0.8\ 0.9\ 0.5]$ (b); $\mathbf{c}=[0.7\ 0.8\ 1\ 0]$ (c) and $\mathbf{c}=[1\ 1\ 1\ 1]$ (d).

example of a two-layer structure, a network of fMUXs is shown in Fig. 3. The characteristics of the network regarded as a function of the select signals x_1 and x_2 are shown in Fig. 4. Depending upon the values of the vector of the information inputs (**c**), we encounter various types of nonlinearities. Fig. 5 includes the corresponding characteristics for the minimum and maximum operations; it becomes apparent that the piecewise character of the relationships becomes predominant.

3. A realization of the network of fMUXs

The functional module of the fuzzy multiplexer introduced above is a generic building block that can be efficiently used to construct larger structures. It functionality is minimal (in the sense we have here only

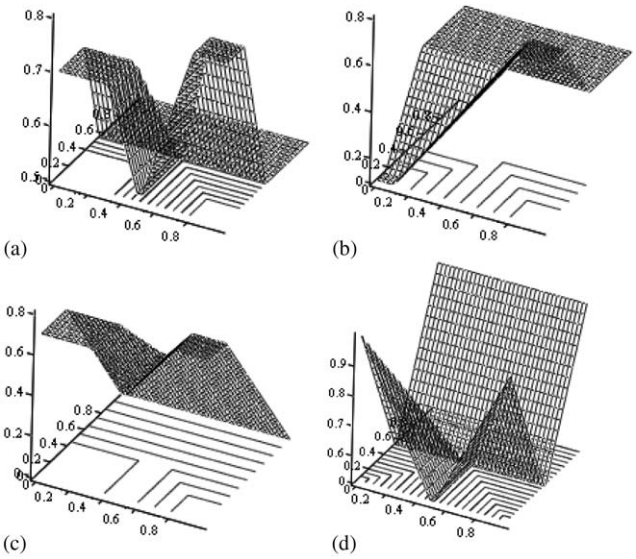

Fig. 5. 3D plots of the characteristics of the fMUX for selected combinations of the parameters and t- and s-norms realized as a minimum and maximum: $\mathbf{c}=[0.7\ 0.8\ 0.9\ 0.5]$ (a); $\mathbf{c}=[0.1\ 0.8\ 0.9\ 0.5]$ (b); $\mathbf{c}=[0.7\ 0.8\ 1\ 0]$ (c) and $\mathbf{c}=[1\ 1\ 1\ 1]$ (d).

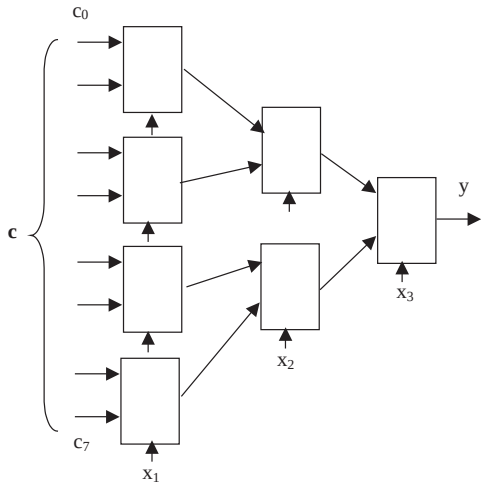

Fig. 6. Network architecture built with the use of the basic functional modules of fuzzy multiplexers; shown are only three layers of the network for illustrative purposes.

a single switching variable, x). If we are dealing with more variables (x_1, x_2, …, x_n), the resulting structure is formed as a regular multilevel architecture composed of the basic fMUXs as visualized in Fig. 6. Noticeably at each level of the network we assign single selection variable. Moreover, at each level of the network the number of multiplexers doubles; the output layer comprises one multiplexer, the layer next to it two multiplexers, the next one four units, etc. With the substantial number of select variables, we can envision some scalability problems (and these have to be addressed at the design phase).

The fMUX network comes with an interesting motivation and exhibits a clear interpretation. As functionality is concerned, it is instructive to go back to the two-valued logic which clearly reveals a rationale behind the use of such networks of multiplexers (McCluskey, 1986). Consider a two-variable Boolean function $f(x_1, x_2)$. According to the classic Shannon (expansion) theorem, the function can be written down as a sum of products, that is

$$y = f(x_1, x_2) = \bar{x}_2 f(x_1, 0) + x_2 f(x_1, 1) = \bar{x}_2[\bar{x}_1 f(0,0) + x_1 f(1,0)] + x_2[\bar{x}_1 f(0,1) + x_1 f(1,1)]. \quad (3)$$

In essence $\mathbf{c} = [f(0,0)\ f(1,0)\ f(0,1)\ f(1,1)]$ becomes here a vector of constant information inputs; these uniquely define the given Boolean function.

This successive expansion of the Boolean function maps directly on the two level structure of the multiplexers; the information inputs to the multiplexer are the functions of one variable, namely $f(x_1,0)$ and $f(x_1,1)$ that are realized by the two multiplexers in the first layer of the network. Here $f(0,0)$, $f(1,0)$, etc. are the information inputs to these multiplexers. The network of the fuzzy multiplexers is just a generalization of this fundamental result to fuzzy functions defined in the unit interval.

When it comes to the interpretation, the network exhibits several interesting properties. We start with the input layer. The outputs of these fMUXs are just logic expressions of a single input (select) variable (being more specific, the variable and its complement). In the sense of involving only one variable they are general. There are also a lot of them (especially if we are dealing with the multiplayer network). In this sense, what becomes realized at the first layer is just a list of partial realizations of the function and the outputs there can be treated as generalized variables. In the subsequent layers these are specialized (by involving another variable) and their number becomes reduced.

4. The general development environment of the network—an architectural layout

The fuzzy multiplexer completes a logic-based processing of input signals and realizes a certain logic-driven mapping between input and output spaces. As they interact with a physical world whose manifestation does not arise at the level of logic (multivalued) signals, it becomes apparent that there is a need for some interface of the model. Such interfaces are well known in fuzzy modeling (Bargiela and Pedrycz, 2002). They commonly arise under a name of fuzzifiers (granular coders) and defuzzifiers (granular decoders). The role of the coder is to convert a numeric input coming from the external environment into the internal format of membership grades of the fuzzy sets defined for each input variable. In a nutshell, this results in a nonlinear normalization of the input (no matter what original ranges the input variables assume) and a linear increase of the dimensionality of the new logic space in comparison with the original one). The decoder takes the results of the logic processing and transforms them into some numeric values. The layered architecture of the fuzzy models with clearly distinguished interfaces and the logic-processing core is illustrated in Fig 7.

With the design of the interfaces, we encounter several main approaches

(a) *Granulation of individual variables:* This mechanism of granulation is quite common in the realm of fuzzy modeling. In essence, we define several fuzzy sets in the universe of discourse of the variable of interest so that any input is transformed via the membership functions defined there and the resulting membership grades are used in further computations by the model. From the design standpoint, we choose a number of fuzzy sets, type of membership functions and a level of overlap between consecutive fuzzy sets. Some general tendencies along this line are thoroughly reported in the literature. By selecting the number of fuzzy sets (usually between 3 and 9), we position modeling activities at some required level of information granularity (a level of modeling details we are interested in). The type of membership functions helps model the semantics of the information granules. Among many possibilities, we commonly encounter triangular fuzzy sets and Gaussian membership functions. These two types come with an extensive list of arguments that help make a suitable selection with respect to the main objectives of the model (e.g., those concerning a tradeoff between interpretation and accuracy of modeling). The overlap level is essential from different points of view, namely (a) semantics of the linguistic terms, (b) nonlinear numeric characteristics of the fuzzy model, and (c) completeness of the model.

(b) *Nonlinear normalization:* Here we transform an original variable defined in some space, say $[a,b]$ (subset of $\mathbf{R}$) is scaled to the unit interval. This could be done by

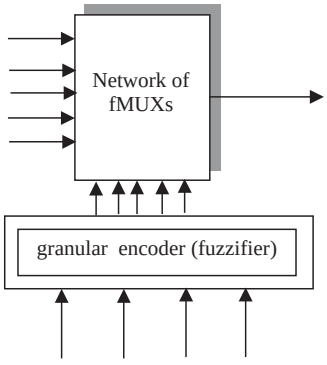

Fig. 7. A general layered structure of fuzzy modeling; the granular decoder is used in case of several networks of fuzzy multiplexers.

a mapping ϕ: $[a,b]\rightarrow[0,1]$ that could be either linear or nonlinear. In any case we consider that ϕ is monotonically increasing with $\phi(a)=0$ $\phi(b)=1$. This transformation does not affect the dimensionality of the problem.

(c) *Simultaneous granulation of many variables:* This technique helps us reduce the dimensionality of the input space (which becomes critical in case of a high number of the variables). The variables are processed at the same time; usually fuzzy clustering is utilized (and this leads to a collection of fuzzy relations rather than fuzzy sets we develop in the first approach). Following clustering (Bezdek, 1981), we depart from using fuzzy *sets* and consider building an interface around fuzzy *relations* defined in a Cartesian product of all variables. The relations are partition matrices formed through fuzzy clustering (such as FCM). We envision the following transformation leading to the fuzzy multiplexing (we assume that the prototypes of the clusters are given, say $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n \in \mathbf{R}^p$):

Given input $\mathbf{x} \in \mathbf{R}^p$; each prototype transforms it into an element in the n-dimensional unit hypercube $\mathbf{u} \in [0,1]^n$ with the coordinates being computed in a well-known manner

$$u_i = \frac{1}{\sum_{j=1}^{n} \left(\|\mathbf{x}-\mathbf{v}_i\|/\|\mathbf{x}-\mathbf{v}_j\|\right)^2}, \tag{4}$$

$i=1, 2, \ldots, p$ (note that we have assumed that the fuzzification factor used in the FCM was set to 2). The optimization of the fMUX network is then carried out by choosing the prototypes (that is building a subspace of the original unit hypercube). As the number of clusters could be adjusted and in general is lower than the original dimensionality of the space (p), this helps us effectively reduce the computational overhead associated with the design of the network.

In this study, while paying attention to the construction of the interfaces, we primarily concentrate on the logic processing and develop a general class of networks based on fuzzy multiplexers.

5. The development of the fMUX networks

In this section, we discuss some general design scenarios and envision their suitability in the development of the fMUX networks.

5.1. Selecting among general design scenarios

The development of the fMUX networks entails two fundamental design scenarios:

(1) If we consider all input variables ($x_1, x_2, \ldots, x_n$) to be used in the development of the system, then a vector of variables $\mathbf{c}=[c_0\ c_1\ c_2\ c_k \ldots]$ has to be estimated.

(2) If the number of the input variables is high (and this implies a high dimensionality of $\mathbf{c}$ along with all drawbacks of learning we envision under such circumstances), the design of the network has to involve a selection of an optimal subset of the variables and a simultaneous estimation of the pertinent vector of constants ($\mathbf{c}$).

In the first scenario, we can use a standard gradient-based learning. It is straightforward; for a given structure (that is the variables being specified in advance along with their arrangement within the network), a detailed form of a performance index to be minimized (Q), specific models of t- and s-norms, the gradient of Q taken with respect to c navigates us through the search space,

$$\mathbf{c}(\text{iter}+1) = \mathbf{c}(\text{iter}) - \beta \nabla_c Q, \tag{5}$$

where $\mathbf{c}$(iter) denotes the values of the input constants at a certain iteration step (iter); $\beta > 0$ is a learning factor implying an intensity of adjustments of the values of $\mathbf{c}$. The gradient of Q can be easily determined. There could be some potential shortcomings of this learning scheme. The most profound one comes with a high dimensionality of the network. If there are a significant number of the variables in the problem, the computed gradient assumes low values. As a result, the learning becomes very inefficient. Note also that the dimensionality of the input vector is equal to 2^n and this expression gives rise to a prohibitively high dimensionality quite quick even for relatively small values of "n". In light of these, it is very likely that the gradient-based methods will come with a limited applicability and we have to proceed with caution when dealing with the increased problem dimensionality.

The second design scenario involves an optimization of the structure (selection of variables) that helps handle the dimensionality problem in an efficient manner. The parametric optimization concerning the vector of the coefficients in some reduced format becomes then more efficient. We may also envision frequent situations in which not all variables become essential to the design of the logic mapping (the same holds in pattern recognition where a stage of feature selection becomes a necessity). With the structural and parametric optimization at hand, we have to confine ourselves to some techniques of global and structrucal optimization. An appealing way to follow is to consider genetic algorithms.

5.2. Genetic development of the fMUX networks

Having recognized the primary design objectives, we now concentrate on the details of the underlying genetic optimization. GAs (Goldberg, 1989, 1991; Michalewicz, 1996) are well documented in the literature along with their numerous applications to neurofuzzy systems; bearing this in mind, we elaborate on the fundamental

architecture of the GA, its parameters and discuss some implementation details.

Genotype representation: The proposed genotype is a direct reflection of the structure of the fMUX network. We consider a floating point coding that results in compact chromosomes. Let us assume that the number of input variables to be used in the network is given in advance and equal to n' where $n' < n$. The chromosome consists of two substrings of real numbers. The first block (substring) contains $2^{n'}$ values of the information inputs (vector). The second block (with n inputs), deals with the subset of the variables to be used in the design. The details are schematically visualized in Fig. 8.

As far as the structure of network is concerned, it is instructive to discuss a way in which the select variables are coded in the second block of the chromosome. The second portion of the chromosome corresponds to the subset of the original inputs that are chosen as select variables and requires some processing before being used to identify the structure of the network. As the entries of the chromosome are real-coded, the likelihood of encountering two identical entries is zero (to be on a safe side, we can always break a tie randomly). With the predefined number of the inputs (n'), we then use only the first n' entries of the chromosome and this produces the sequence of the input variables. The entries are ranked (in the increasing order) and the first n' entries of the substring are used to choose among all variables. This ordering is directly mapped onto the network where the first variable is the one switching the first layer of the network.

N.B. One could easily optimize the number of the subset of the input variables (n') instead of supplying it externally; yet this does not to seem to be very attractive. It is perhaps more justifiable to do a systematic search by sweeping n' from 1 to n. In essence, this systematic search helps us assess approximation and generalization abilities of the networks and get a better sense as to the plausible subset of the variables.

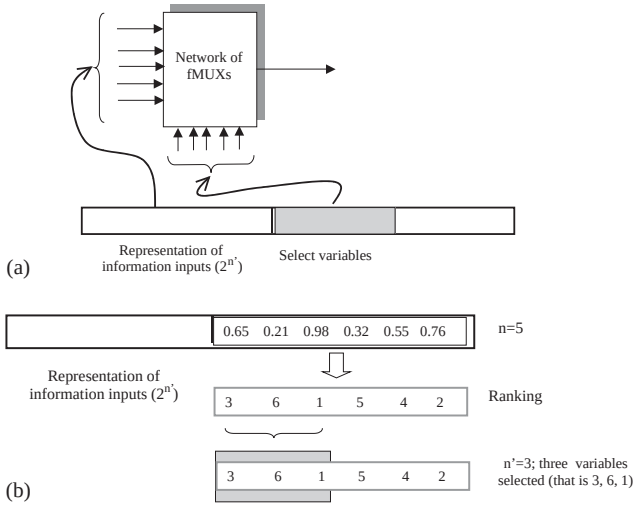

Fig. 8. The structure of the fMUX network and its genetic representation (a) and details of the coding of the subset of the input variables (b) through ranking and using the first n' entries of the substring.

The basic mechanisms of genetic optimization involve selection process quite commonly using an elitist ranking selection (Baker, 1985), mutation, and crossover (e.g., BLX-0.5, (Eshelman and Schaffer, 1993; Herrera et al., 1998)). The fitness function quantifies how the network approximates the data and is taken as $1 - Q/(Q + \varepsilon)$, with Q being a sum of squared errors between the target values (experimental output data) and the corresponding outputs of the network. A small positive constant ε standing in the denominator of the above expression assures that the fitness function remains meaningful even for $Q = 0$ (which in practice never occurs).

6. Numeric illustration

The experiments reported in this section are intended to illustrate the development, performance, and interpretation issues of the proposed network. Having these in mind, we discuss three categories of data. The first are just Boolean (binary) data; in this case we are interested to learn about the performance of the fMUX so the results could be easily contrasted with those obtained using "standard" design techniques encountered in digital logic. The second one deals with a one-dimensional input–output synthetic data; the low dimensionality helps to visualize the details of the network. Finally, we consider an auto-mpg data set coming from the Machine Learning repository (Merz and Murphy, 1998). In the series of the experiments, these two parameters are fixed with 100 or 200 individuals in a population and between 200 and 500 generations. These values were experimentally selected; the number of generations is more than sufficient.

6.1. Realization of boolean functions

In this experiment, we are concerned with Boolean data; this helps us compare the result produced by the fuzzy multiplexer with the solutions obtained using standard techniques used to design digital systems. The data set comprises of 12 input–output pairs of binary data with 5 inputs and a single output,

$$(\mathbf{x}(k)\text{-}y(k)) : ([10000]0), ([10001]1), ([10010]1), ([10011]1), ([11100]1), ([11101]0), ([11110]1), ([11111]1), ([11000]0), ([11001]1), ([11010]0), ([11011]0).$$

The development of the network is completed for a varying number of inputs starting from one variable and

Table 1
Structure of the multiplexer network and associated errors

Number of variables	1	2	3	4	5
Order of variables	x_3	x_5, x_3	x_3, x_5, x_2	x_4, x_3, x_2, x_5	x_4, x_1, x_2, x_5, x_3
MSE	0.478	0.408	0.354	$1.29*10^{-4}$	$2.09*10^{-5}$
Classification error	3	2	4	0	0

By the classification error we mean the number of mismatches between the binary (that is thresholded) output of the network and the data.

ending up with all variables. The results are summarized in Table 1.

From this table (based on the values of the classification error), it becomes obvious that 4 variables are optimal for the problem. The resulting inputs are shown in Fig. 9. With the threshold of 0.5, we are left with eight significant inputs. Noticeably, all those very close to 1 identify the minterms existing in the data. The one extra here with the value equal to 0.61 (and still deemed relevant) corresponds to the combination of the inputs equal to 0100 (that is $\bar{x}_4 x_3 \bar{x}_2 \bar{x}_5$) and it is subsumed by the remaining sum of the minterms. Alluding to the problem variables being identified by the genetic algorithm, it is interesting to note that x_1 has been eliminated. It is not surprising at all noting that it fixed at 1 and thus becomes redundant in the problem. For this optimal number of the inputs, Fig. 10 shows how GA performs in terms of the optimization; evidently most learning occurs at the beginning of the process and this becomes evident for the best individual as well as the average fitness in the population.

The second example is shown here to visualize the effectiveness of the genetic optimization. The binary data describe a three dimensional XOR problem. With the population of 50 chromosomes, 50 generations, the mutation rate of 0.05, and crossover rate equal to 0.8 the ideal result is obtained after a few initial generations. As expected, the information inputs are either close to 1 or become practically equal to zero, Fig. 11.

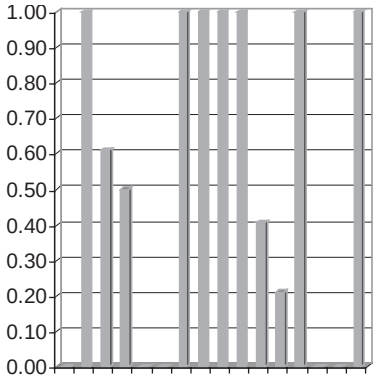

Fig. 9. Indexes of the information inputs; observe that their distribution is highly bimodal with the values of information inputs being close to 1 or 0 with a few exceptions.

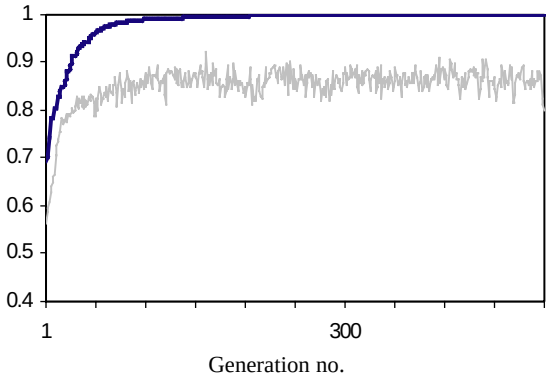

Fig. 10. Fitness function (average and best individual) in successive generations.

6.2. Auto miles per gallon data

This data set comes from the Machine Learning repository and concerns relationships between the characteristics of vehicles (weight, displacement, number of cylinders, etc) and their fuel consumption (expressed in miles per gallon).

For purposes of this experiment and considering that we have a number of input variables, we build new normalized variables; see Section 4. The genetic optimization was completed for 200 individuals and run for 600 generations. Table 2 summarizes the performance of the network for the genetically optimized subsets of input variables. The resulting optimal information inputs are shown in Fig. 12. Noticeably a significant number of them are quite low (that is assuming values below the 0.5 threshold level).

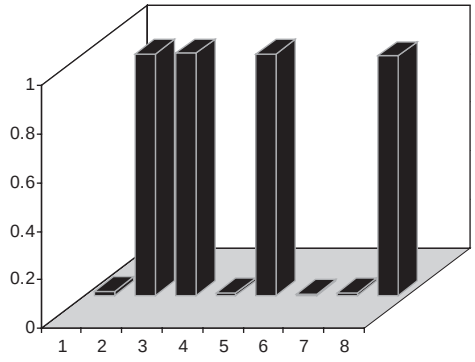

Fig. 11. Information inputs of the network of fuzzy multiplexers.

The "optimal" structure includes only a few input variables. For these we have a number of "meaningful" logic combinations of the select variables. The approx-

Table 2
Performance of the fMUX network for the optimal subsets of the input variables

Number of input variables	1	2	3	4	5	6	7
Variables	Weight	Weight, year of model	Weight, displacement, year of model	Displacement, horsepower, weight, year of model	Horsepower, year of model, weight, origin, displacement	Displacement, origin, year of model, number of cylinders, horsepower, weight	Origin, year of model, displacement, acceleration, number of cylinders, horsepower, weight
Performance index (training set)	0.113	0.086	0.076	0.074	0.078	0.090	0.112
Performance index (testing set)	0.122	0.089	0.081	0.078	0.082	0.101	0.124

imate logic expression reads as follows:

fuel consumption = (¬(displacement) *and* ¬(horsepower) *and* ¬(weight) *and* year_of_model) *or* (¬(displacement) *and* horsepower *and* ¬(weight) *and* year_of_model) *or* (displacement *and* ¬(horsepower) *and* weight *and* year of model) *or* (¬(displacement) *and* ¬(horsepower) *and* ¬(weight) *and*¬(year of model).

Let us recall that considering the nature of the transformation, the negation symbol (¬) showing with some variables indicates that the decrease in the values of such variables implies increase in the output variable. The intuitive meaning of the logic description of the data is quite straightforward; e.g., the first combination of the variables indicates that lower displacement *and* lower horsepower *and* lower weight of vehicle *and* most recent model imply higher fuel efficiency.

7. Conclusions

We have introduced a logic-driven architecture of fuzzy models based on the concept of fuzzy multiplexers (fMUXs). fMUXs are direct generalizations of fundamental building blocks encountered in two-valued (digital) logic and being used in a design process therein. The design of the fMUX networks has been carried in the framework of genetic optimization. In this study, the GA is aimed both at the structural and parametric optimization. It is worth stressing that the structural optimization becomes indispensable in case of multivariable problems. The selected (optimized) subset of input variables leads to an efficient dimensionality reduction and helps concentrate on the most significant variables. The transparency of the model is also worth emphasizing; the network is an immediately interpretable construct that is translated into a coherent logical description of data. We have emphasized the role of the interface layer of a fuzzy model and shown that it is directly related to the level of detail we would like to capture when developing the logic description of data (model). By increasing the number of linguistic landmarks (fuzzy sets) defined in the input spaces, we end up with more detailed logic description of data; however, some of the descriptors (terms) produced by the individual fMUXs may not be highly relevant (that is associated with the low entries of (**c**).

The experimental studies have been conducted for several categories of data (problems) starting from Boolean data (that helped us discuss the fMUX networks vis-à-vis digital logic design) and ending up

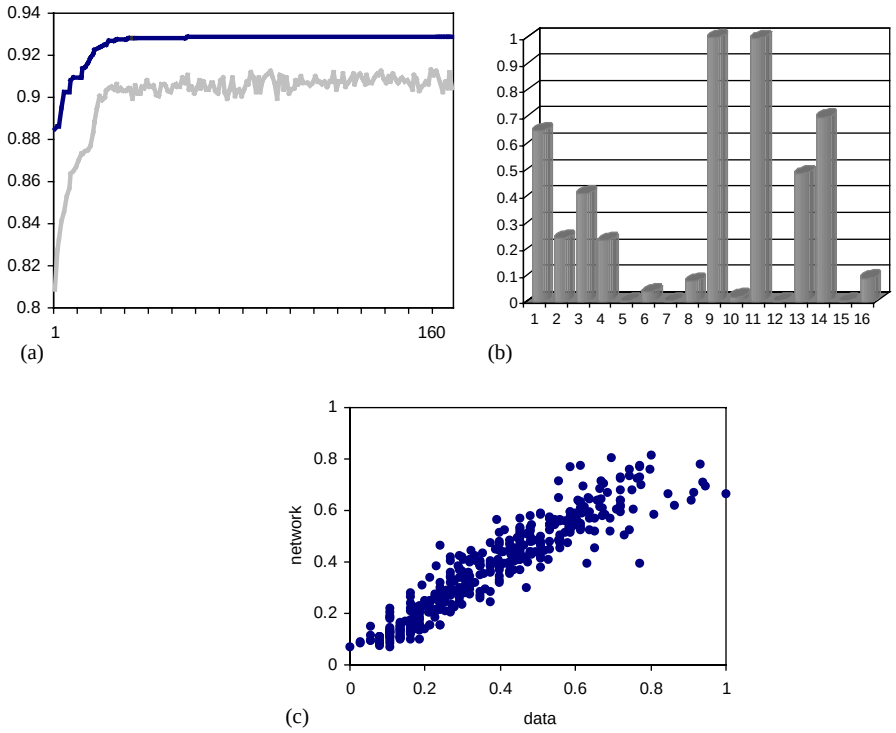

Fig. 12. GA optimization—best individual and average fitness function (a) values of the information inputs (b), and data versus results produced by the network (c).

with one of the Machine Learning datasets. As our primary intent was to focus on the logic-driven facet of fuzzy modeling and interpretability aspects of the network, we have decided not to proceed with a comprehensive comparative analysis with other fuzzy models that might not be that easily amenable to the logic-inclined interpretation.

Acknowledgements

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

References

Baker, J.E., 1985. Adaptive selection methods for genetic algorithms. Proceedings of the First International Conference on Genetic Algorithms, pp. 101–111.

Bargiela, A., Pedrycz, W., 2002. Granular Computing: An Introduction. Kluwer Academic Publishers, Dordrecht.

Bezdek, J.C., 1981. Pattern Recognition with Fuzzy Objective Functions. Plenum, NewYork.

Ciletti, M.D., 1999. Modeling, Synthesis and Rapid Prototyping with the Verilog HDL. Prentice-Hall, Upper Saddle River, NJ.

Delgado, M., Gomez-Skarmeta, A.F., Martin, F., 1997. A fuzzy clustering-based prototyping for fuzzy rule-based modeling. IEEE Transactions on Fuzzy Systems 5 (2), 223–233.

Eshelman, L.J., Schaffer, J.D., 1993. Real-coded Genetic algorithms and interval schemata. In: Foundations of Genetic Algorithms, Vol. 2. Morgan Kaufman Publishers, San Mateo, CA, pp. 187–202.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA.

Goldberg, D.E., 1991. Real-coded genetic algorithms, virtual alphabets, and blocking. Complex Systems 5, 139–167.

Gomez-Skarmeta, A.F., Delgado, M., Vila, M.A., 1999. About the use of fuzzy clustering techniques for fuzzy model identification. Fuzzy Sets and Systems 106, 179–188.

Herrera, F., Lozano, M., Verdegay, J.L., 1998. Tackling real-coded genetic algorithms: operators and tools for behavioral analysis. Artificial Intelligence Review 12, 256–319.

Kohavi, Z., 1970. Switching and Finite Automata Theory. McGraw-Hill, NewYork.

Mano, M.M., 1991. Digital Design 2nd Edition. Prentice-Hall, Englewood Cliffs, NJ.

McCluskey, E.J., 1986. Logic Design Principles. Prentice-Hall, Englewood Cliffs, NJ.

Merz, C.J., Murphy, P.M., 1998. UCI repository of machine learning databases. Technical Report, Department of Information and Computer Science, University of California at Irvine, [http://www.ics.uci.edu/~mlearn/MLRepository.html].

Michalewicz, Z., 1996. Genetic Algorithms + Data Structures = Evolution Programs, third ed. Springer, Heidelberg.

Zadeh, L.A., Kacprzyk, J. (Eds.), 1999. Computing with Words in Information/Intelligent Systems, vols. I and II. Physica Verlag, Heidelberg.