CHAPTER 10

NEURAL ARCHITECTURES OF FUZZY PETRI NETS

W. Pedrycz
Department of Electrical & Computer Engineering
University of Alberta, Edmonton
Canada T6G 2G7
&
Systems Research Institute
Polish Academy of Sciences
Warsaw, Poland
pedrycz@ee.ualberta.ca

In this chapter, we discuss a novel approach to pattern classification using a concept of fuzzy Petri nets. In contrast to the commonly encountered Petri nets with their inherently Boolean character of processing tokens and firing transitions, the proposed generalization involves continuous variables. This extension makes the nets to be fully in rapport with the panoply of the real-world classification problems. The introduced model of the fuzzy Petri net hinges on the logic nature of the operations governing its underlying behavior. The logic-driven effect in these nets becomes especially apparent when we are concerned with the modeling of its transitions and expressing pertinent mechanisms of a continuous rather than an on-off firing phenomenon. An interpretation of fuzzy Petri nets in the setting of pattern classification is provided. This interpretation helps us gain a better insight into the mechanisms of the overall classification process. Input places correspond to the features of the patterns. Transitions build aggregates of the generic features giving rise to their logical summarization. The output places map themselves onto the classes of the patterns while the marking of the places correspond to the class of membership values. Details of the learning algorithm are also provided along with an illustrative numeric experiment.

1 Introduction

In recent years, Petri nets [5] have started to gain in importance in the areas of knowledge representation, robot planning, and expert systems, see, for instance, [1], [2], [3], and [4]. Surprisingly, little research has been done on the use of Petri net in pattern classification. On the other hand, most classification pursuits are easily formalized in the setting of Petri nets once these architectures become generalized in a way that they reflect a continuous character omnipresent in most of the classification tasks. The approach taken here dwells on the existing fuzzy set-based augmentation of the generic version of Petri nets [6], [7]. Fuzzy sets contribute not only to a way in which an issue of partial firing of the transitions can be addressed but they provide a significant level of parametric flexibility. This flexibility becomes indispensable in the case of training of such fuzzy Petri nets – the feature not being available in their standard binary counterparts. The objective of this study is to investigate fuzzy Petri nets in the framework of pattern recognition and make them conceptually and computationally viable as pattern classifiers.

The material of the chapter is arranged into 7 sections. We start off with a brief introduction to Petri nets along with their fuzzy set-based generalization. This generalization helps us capture and formalize the notion of continuous rather than straight on-off firing mechanism. In Section 3, we analyze the details of the fuzzy Petri net providing all necessary computational details. Subsequently, Section 4 deals with a process of learning in the nets. The main objective of such learning is to carry out some parametric optimization so that the network can adjust to the required training set of patterns (being composed of pairs of marking of input and output places). A way of interfacing fuzzy Petri nets with the modeling environment is discussed in Section 5. Numerical experiments are reported in Section 6 while conclusions are covered in Section 7.

2 The Generalization of the Petri Net and Its Underlying Architecture

Let us briefly recall the basic concept of a Petri net. Formally speaking, a Petri net [4], [5] is a finite graph with two types of nodes, known as places (P) and transitions (T). More formally, the net can be viewed as a triple (P, T, F) where

$$P \cap T = \varnothing$$

$$P \cup T \neq \varnothing$$

$$F \subseteq (P \times T) \cup (T \times P)$$

$$\text{domain}(F) \cup \text{codomain}(F) = P \cup T$$

In the above, F is called the flow relation. The elements of F are the arcs of the Petri net.

Each place comes equipped with some tokens that form a marking of the Petri net. The flow of tokens in the net occurs through firings of the transitions; once all input places of a given transition have a nonzero number of tokens, this transition fires. Subsequently, the tokens are allocated to the output places of the transition. Simultaneously, the number of tokens at the input places is reduced. The effect of firing of the transitions is binary: the transition either fires or does not fire.

An important generalization of the generic model of the Petri net is to relax the Boolean character of the firing process of the transition. Instead of subscribing to the firing-no firing dichotomy, we propose to view the firing mechanism as a gradual process with a continuum of possible numeric values of the strength (intensity) of firing of a given transition. Then the flow of tokens can also take this continuity into consideration meaning that we end up with the marking of the places that become continuous as well. Evidently, such a model is in rapport with a broad class of real-world phenomena including pattern classification. The generalization of the net along this line calls for a series of pertinent realization details. In what follows, we propose a construct whose functioning adheres as much as possible to the logic fabric delivered by fuzzy sets. In this case, a sound solution is to adopt

the ideas of fuzzy logic as the most direct way of implementation of such networks.

3 The Architecture of the Fuzzy Petri Net

The topology of the fuzzy Petri net as being cast in the framework of pattern classification is portrayed in Figure 1. As it will be shown further on, this setting nicely correlates with the classification activities encountered in any process of pattern recognition.

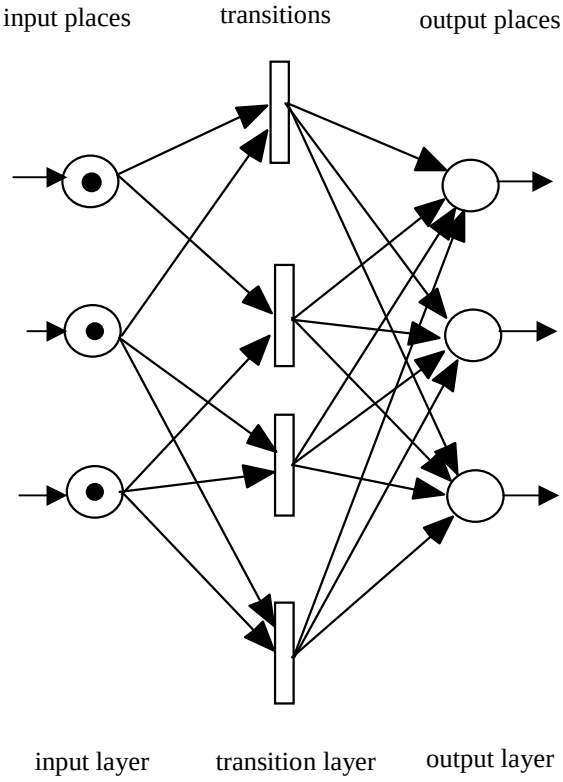


Figure 1. A general three layer topology of the fuzzy Petri net.

The network constructed in this manner comprises three layers:

- an input layer composed of “n” input places;
- a transition layer composed of “hidden” transitions;
- an output layer consisting of “m” output places.

The input place is marked by the value of the feature (we assume that the range of the values of each of the features is in the unit interval). These marking levels are processed by the transitions of the network whose levels of firing depend on the parameters associated with each transition such as their threshold values and the weights (connections) of the incoming features. Subsequently, each output place corresponds to a class of patterns distinguished in the problem. The marking of this output place reflects a level of membership of the pattern in the corresponding class.

The detailed formulas of the transitions and output places rely on the logic operations encountered in the theory of fuzzy sets. The i-th transition (more precisely, its activation level z_i) is governed by the expression

$$z_i = \mathop{T}_{j=1}^{n}[w_{ij} s(r_{ij} \rightarrow x_j)], \qquad j = 1, 2, \ldots, n; \; i = 1, 2, \ldots, \text{hidden},$$

where:

- w_{ij} is a weight (connection) between the i-th transition and the j-th input place;

- r_{ij} is a threshold level associated with the level of marking of the j-th input place and the i-th transition; and

- the level of marking of the j-th input place is denoted by x_j

Moreover, "t" is a t-norm, "s" denotes an s-norm, while $\rightarrow$ stands for an implication operation expressed in the form

$$a \rightarrow b = \sup\{c \in [0,1] \mid atc \leq b\} \tag{1}$$

where a, b are the arguments of the implication operator confined to the unit interval. Note that the implication is induced by a certain t-norm. In the case of two-valued logic, (1) returns the same truth values as the commonly known implication operator, namely

$$a \rightarrow b = \begin{cases} b, \text{if } a > b \\ 1, \text{otherwise} \end{cases} = \begin{cases} 0, \text{if } a = 1 \text{ and } b = 0 \\ 1, \text{otherwise} \end{cases} \qquad a, b \in \{0,1\}$$

The j-th output place (more precisely, its marking y_j) summarizes the levels of evidence produced by the transition layer and performs a nonlinear mapping of the weighted sum of the activation levels of these transitions (z_i) and the associated connections v_{ji}

$$y_j = f\left(\sum_{i=1}^{\text{hidden}} v_{ji} z_i \right), \qquad j = 1, 2, \ldots, m \tag{2}$$

where "f" is a nonlinear monotonically increasing function (mapping) from **R** to [0,1].

The role of the connections of the output places is to modulate an impact the firing of the individual transition exhibits on the accumulation of the tokens at this output place (viz. the membership value of the respective class). The negative values of the connections have an inhibitory effect meaning that the value of the class membership becomes reduced.

Owing to the type of the aggregation operations being used in the realization of the transitions, their interpretation sheds light on the way in which the individual features of the problem are treated. In essence, the transition produces some higher level, synthetic features out of these originally encountered in the problem and represented in the form of the input places. The weight (connection) expresses a global contribution of the j-th feature to the i-th transition: the lower the value of w_{ij}, the more significant the contribution of the feature to the formation of the synthetic aggregate feature formed at the transition level. The connection itself is weighted uniformly regardless of the numeric values it assumes. The more selective (refined) aggregation mechanism is used when considering threshold values. Referring to (1), one easily finds that the thresholding operation returns 1 if x_j exceeds the value of the threshold r_{ij}. In other words, depending on this level of the threshold, the level of marking of the input place becomes "masked" and the threshold operation returns one. For the lower values of the marking, such levels are processed by the implication operation and contribute to the overall level of the firing of the transition.

One should emphasize that the generalization of the Petri net proposed here is in full agreement with the two-valued generic version of the net commonly encountered in the literature. Consider, for instance, a single

transition (transition node). Let all its connections and thresholds be restricted to {0, 1}. Similarly, the marking of the input places is also quantified in a binary way. Then the following observations are valid:

- only those input places are relevant to the functioning of the i-th transition for which the corresponding connections are set to zero and whose thresholds are equal to 1. Denote a family of these places by $\mathbf{P}$

- the firing level of the i-th transition is described by the following formula,

$$z_i = \underset{j \in P}{\overset{n}{T}} (r_{ij} \rightarrow x_j)$$

It becomes apparent that the firing level is equal to 1 if and only if the marking of all input places in $\mathbf{P}$ assume 1; the above expression for the transition is nothing but an *and*-combination of the levels of marking of the places in $\mathbf{P}$, namely

$$z_i = \underset{j \in P}{\overset{n}{T}} x_j$$

(let us recall that any t-norm can be used to model the *and* operation; moreover all t-norms are equivalent when operating on the 0-1 truth values).

4 The Learning Procedure

The learning completed in the fuzzy Petri net is the one of parametric nature, meaning that it focuses on changes (updates) of the parameters (connections and thresholds) of the net. The structure of the net is left unchanged. These updates are carried in such a way so that a certain predefined performance index becomes minimized. To help concentrate on the detailed derivation of the learning formulas, it is advantageous to view a fully annotated portion of the network as illustrated in Figure 2.

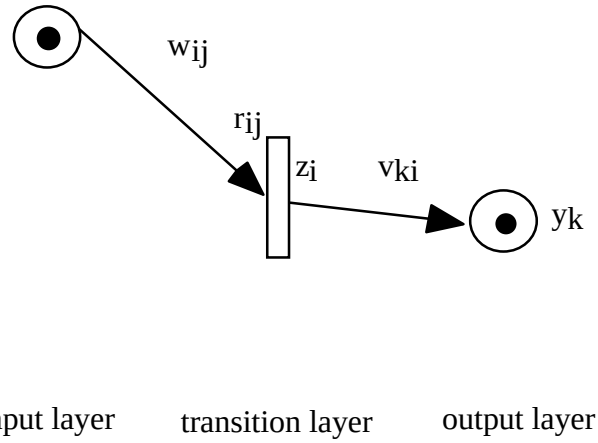


Figure 2. Optimization in the fuzzy Petri net; a section of the net outlines all notation being used in the learning algorithm.

The performance index to be minimized is viewed as a standard sum of squared errors. The errors are expressed as differences between the levels of marking of the output places of the network and their target values. The considered on-line learning assumes that the modifications to the parameters of the transitions and output places occur after presenting an individual pair of the training sets, say marking of the input places (denoted by **x**) and the target values (namely, the required marking of the output places) expressed by **t**. Then the performance index for the input-output pair reads as

$$Q = \sum_{k=1}^{m} (t_k - y_k)^2 \tag{3}$$

The updates of the connections are governed by the standard gradient-based method

$$\mathbf{param}(\text{iter}+1) = \mathbf{param}(\text{iter}) - \alpha \nabla_{\mathbf{param}} Q \tag{4}$$

where $\nabla_{\mathbf{param}} Q$ is a gradient of the performance index Q taken with respect to the parameters of the fuzzy Petri net. The iterative character of the learning scheme is underlined by the parameter vector regarded as a function of successive learning epochs (iterations).

The intensity of learning is controlled by the positive learning rate (α). In the above scheme, the vector of the parameters, **param**, is used to encapsulate all the elements of the structure to be optimized. Further on they will be made more detailed, as we will proceed with a complete description of the update mechanism. With the quadratic performance index (3) in mind, the following holds

$$\nabla_{\mathbf{param}} Q = -2\sum_{k=1}^{m} (t_k - y_k)\ \nabla_{\mathbf{param}} y_k$$

Moving into detailed formulas refer again to Figure 2. Moreover, the nonlinear function associated with the output place (2) is a standard sigmoid nonlinearity described as

$$y_k = \frac{1}{1+\exp(-z_k)}$$

For the connections of the output places we obtain

$$\frac{\partial y_k}{\partial v_{ki}} = y_k(1-y_k)z_i$$

k=1, 2, …,m, i=1, 2, …, hidden. Observe that the derivative of the sigmoidal function is equal to $y_k(1 - y_k)$.

Similarly, the updates of the threshold levels of the transitions of the net are expressed in the form

$$\frac{\partial y_k}{\partial r_{ij}} = \frac{\partial y_k}{\partial z_i}\frac{\partial z_i}{\partial r_{ij}} \qquad i=1, 2, \ldots,n.$$

In the sequel, we obtain

$$\frac{\partial y_k}{\partial z_i} = y_k(1-y_k)v_{ki}$$

and

$$\frac{\partial z_i}{\partial r_{ij}} = A\frac{\partial}{\partial r_{ij}}(w_{ij} + (r_{ij} \to x_j) - w_{ij}(r_{ij} \to x_j)) = A(1-w_{ij})\frac{\partial}{\partial r_{ij}}(r_{ij} \to x_j)$$

where the new expression, denoted by A, is defined by taking the t-norm over all the arguments but "j",

$$A = \underset{\substack{l=1 \\ l \neq j}}{\overset{n}{T}}[w_{il} s(r_{il} \rightarrow x_l)]$$

The calculations of the derivative of the implication operation can be completed once we confine ourselves to some specific realization of the t-norm that is involved in its development. For the product (being a particular example of the t-norm), the detailed result reads as

$$\frac{\partial}{\partial r_{ij}}(r_{ij} \rightarrow x_j) = \frac{\partial}{\partial r_{ij}} \begin{cases} x_j / r_{ij}, \text{ if } r_{ij} > x_j \\ 1, \text{otherwise} \end{cases} = \begin{cases} -x_j / r^2_{ij}, \text{ if } r_{ij} > x_j \\ 0, \text{ otherwise} \end{cases}$$

The derivatives of the connections of the transitions (transition nodes) are obtained in a similar way. We get

$$\frac{\partial y_k}{\partial w_{ij}} = \frac{\partial y_k}{\partial z_i} \frac{\partial z_i}{\partial w_{ij}} \qquad k=1, 2, \ldots,m, i=1, 2, \ldots,\text{hidden}, j=1, 2,\ldots,n$$

Subsequently, one derives

$$\frac{\partial z_i}{\partial w_{ij}} = A \frac{\partial}{\partial w_{ij}}(w_{ij} + (r_{ij} \rightarrow x_j) - w_{ij}(r_{ij} \rightarrow x_j)) = A(1 - (r_{ij} \rightarrow x_j))$$

There are two aspects of further optimization of the fuzzy Petri nets that need to be raised in the context of their learning:

- the number of nodes in the transition layer. The optimization of the number of the transition nodes of the fuzzy Petri net falls under the category of structural optimization that cannot be handled by the gradient-based mechanisms of the parametric learning. By increasing the number of these nodes, we enhance the mapping properties of the net as each transition can be fine-tuned to fully reflect the boundaries between the classes. Too many of these transitions, however, could easily develop a memorization effect that is well-known in neural networks.

- the choice of specific t-norm and s-norm. This leads us to an aspect of a semi-parametric optimization of the fuzzy Petri net. The choice of these norms does not impact the architecture of the net; yet in this optimization we cannot resort ourselves to the gradient-based learning. A prudent way to follow would be to confine to a family of t-norms and s-norms that can be systematically exploited one by one.

5 Interfacing Fuzzy Petri Nets with Granular Information

In addition to the design of the fuzzy Petri net, we are also concerned with its interfacing of the environment in which it has to perform. It is accomplished by defining a certain functional module that helps transform physical entities coming from the environment into more abstract and logic-inclined entities to be used directly by the fuzzy Petri net. The essence of such interfaces is in information granulation and the use of such granules as a bridge between a numeric information generated by the physical environment and the logical layer of information granules visible at the level of the Petri net itself. In what follows, we elaborate on these two phases in more detail:

(i) *Construction of information granules.* The granulation of information can be carried out in different ways. The use of Fuzzy C-Means or other fuzzy clustering technique is a viable way to follow. Let us briefly recall that the crux of clustering is to form information granules – fuzzy sets (fuzzy relations) – when starting from a cloud of numeric data. The result of clustering comes in the form of the prototypes of the clusters and a fuzzy partition summarizing a way the entire data set becomes divided (split) into clusters. As the name stipulates, the partitioning assigns elements to different clusters to some degree with the membership values between 0 and 1. For "c" clusters we end up with "c" prototypes, $\mathbf{v}_1$, $\mathbf{v}_2$, …, $\mathbf{v}_c$.

(ii) *Determination of the activation levels of information granules.* Any new input **x** "activates" the i-th cluster according to the formula

$$u_i(\mathbf{x}) = \frac{1}{\sum_{i=1}^{c} \frac{\| \mathbf{x} - \mathbf{v}_i \|^2}{\| \mathbf{x} - \mathbf{v}_j \|^2}}$$

where ||.|| is a distance function defined between **x** and the respective prototype. The calculations shown above form a core of the interface structure of the fuzzy Petri net, see Figure 3. The activation levels are directly used as the marking of the input places.

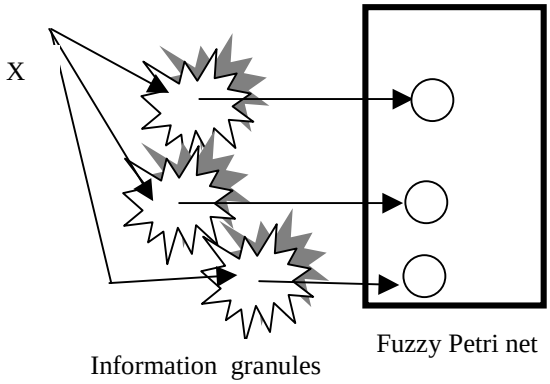


Figure 3. Interfacing the world with the fuzzy Petri net.

An important issue arises when the input (**x**) is not numeric but comes as some less specific information granule. In particular, we may anticipate a granular information represented in the form of a certain numeric interval, see Figure 4.

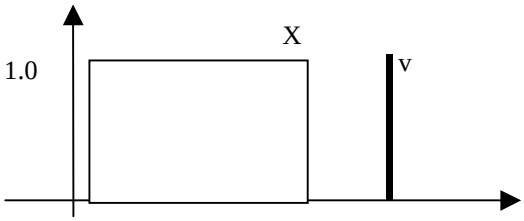


Figure 4. An interval-based information granule and distance calculations.

Taking into consideration the interval-based information granule, the distance function needs some modification. The calculations are carried out in the form

$$\| X - v_i \| = \min_{z \in X} \| z - v_i \|$$

where X is the numeric granule. (Note that the above formula deals with a certain coordinate of **x** and prototype **v**, say x and v.) These computations provide us with an optimistic (that is minimal) value of the distance function. This, in turn, activates the linguistic granules to a higher extent in comparison to what it would have been in the pure numeric case. Subsequently, the activation levels sum up to the value higher than 1 (recall that this sum of the activation levels is always equal to one in the case of numeric inputs coming from the modeling environment). In this sense this sum (or its departure from one) can serve as a useful indicator of the granularity level of the input information.

To quantify this observation, we discuss a one-dimensional case and consider three prototypes located at 1.0, 4.0, and 7.0, respectively. The nonnumeric information is represented as an interval distributed around x with the bounds located at x–d and x+d.The distance is computed as discussed above. The plots of the activation of the first information granule distributed around 1.0 are illustrated in Figure 5.

Subsequently, Figure 6 visualizes a sum of the activation levels of the linguistic granules implied by the interval-valued input. Observe that with the increase of "d", the sum starts to exceed 1.

6 Experiments

In this section, we discuss some experimental results. For illustrative purposes we focus on the classification problems that involve only two features.

Experiment 1. The patterns themselves are generated by two fuzzy functions

$$f_1(x_1, x_2) = (\overline{x_1} t x_2) s (x_1 t \overline{x}_2)$$

$$f_2(x_1, x_2) = (x_1 t 0.2) s (\overline{x}_2) \qquad (5)$$

In these two functions, the t- and s-norms are implemented using the product (atb=ab) and probabilistic sum (asb=a+b–ab), respectively, a,b∈[0,1]. The overbar denotes a complement of the feature's value,

$\overline{x} = 1 - x$, $x \in [0,1]$. Note that the first function is a multivalued (fuzzy) Exclusive-Or function (XOR) whereas the second one assumes its high truth values at the upper corner of the unit square. The values assumed by these two functions are treated as continuous class memberships. The two variables of the functions form the features. The plots contained in Figures 7 and 8 portray the functions themselves as well as the classification boundaries occurring between the classes (viz. the curves along which the membership grades in two classes are equal).

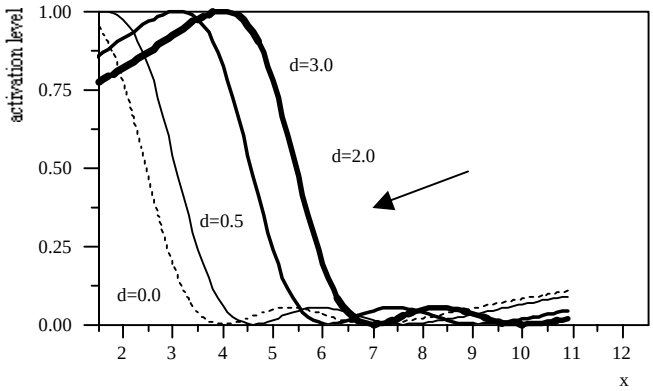


Figure 5. Activation level (membership grade) of the first prototype regarded as a function of x for selected values of information granularity (d).

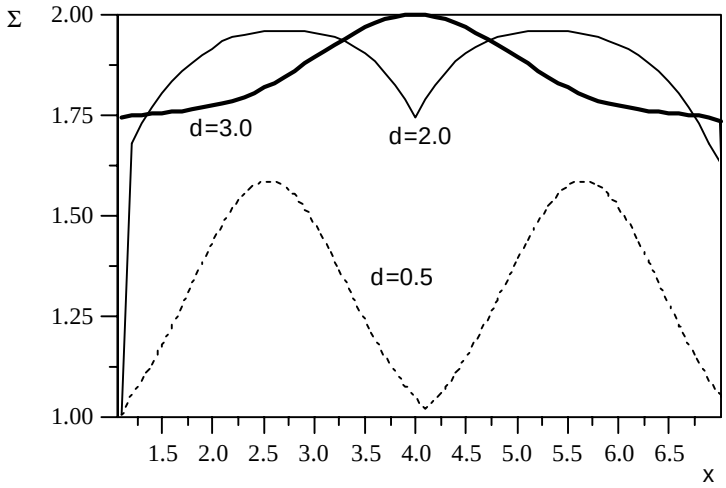
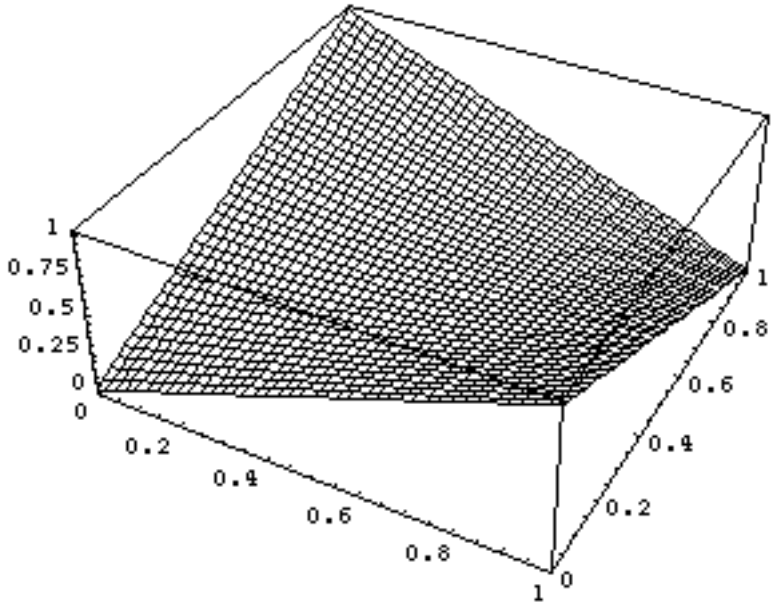
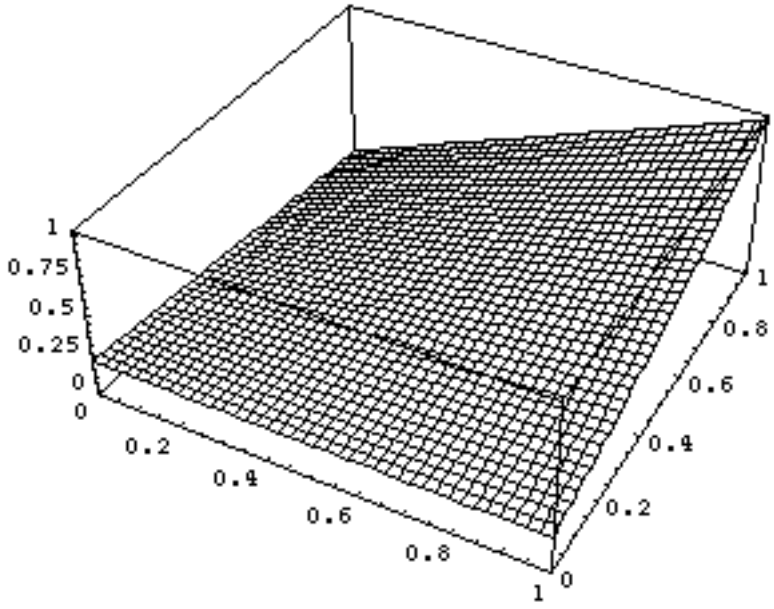


Figure 6. Sum of activation levels (Σ) of the linguistic granules for some selected values of "d".



(a)



(b)

Figure 7. Plots of the two logic functions used in the experiment (a) f_1 and (b) f_2.

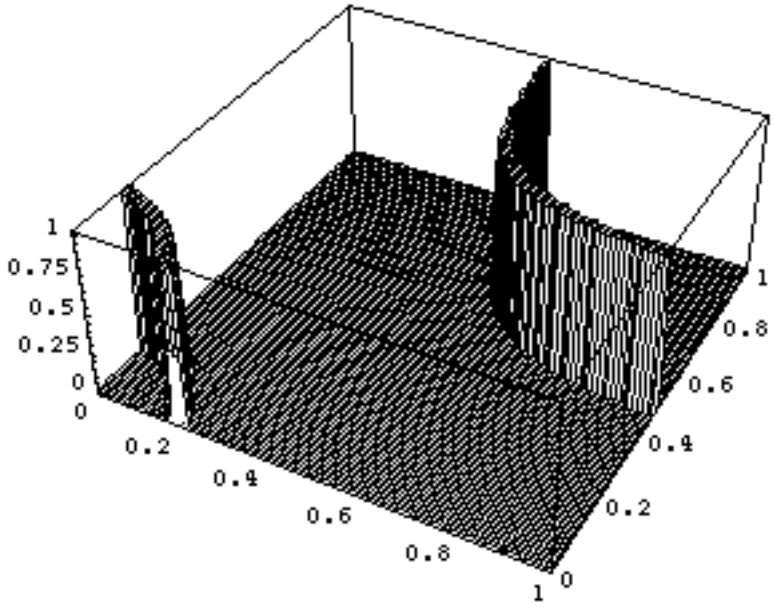


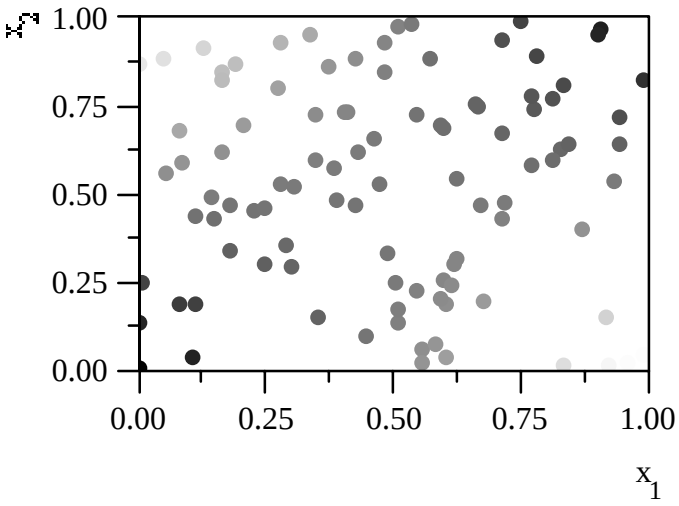
Figure 8. Classification boundaries between the classes.

Note that the problem itself is nonlinear; furthermore the classification boundaries give rise to the disjoint classification regions.

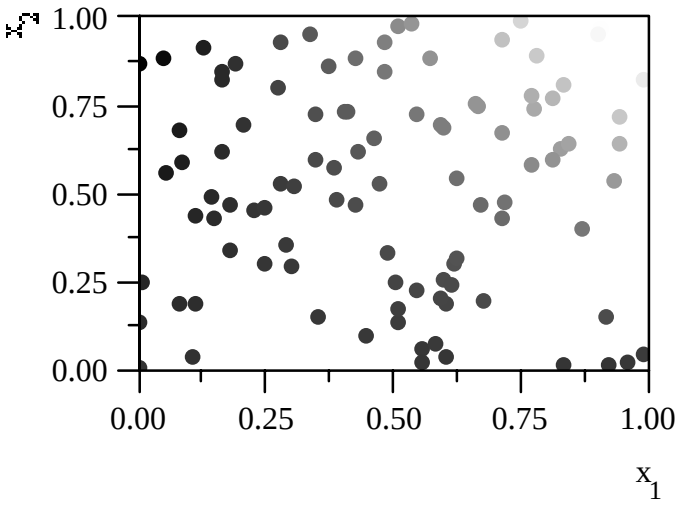
Let us now set up a topology of the fuzzy Petri net. The network has four input places. These correspond to the two original features and their complements. The output layer consists of two output places reflecting the number of the classes occurring in the classification problem. The number of transitions (viz. the size of the transition layer, denoted by "hidden") varies from 2 to 8 throughout the topologies of the Petri nets. The learning was completed in an on-line version meaning that the updates of the connections (parameters) of the Petri net are carried out for each input-output pair of the training data. The experiment involves a standard training-testing scenario: the training was done based on 100 patterns generated randomly (over $[0,1]\times[0,1]$) from the model (5); the testing set involves another 100 patterns again governed by (5). The corresponding data sets are illustrated in Figures 9 and 10.

The learning rate (α) was set to 0.02; the intent was to assure a stable learning process not allowing for any oscillations. Obviously, one can increase the value of the learning rate and therefore accelerate learning

without sacrificing its stability. Nevertheless, the issue of efficiency of learning was not a primary concern in this experiment. This is particularly so, as the learning itself is rather fast and does not require a significant number of learning epochs.

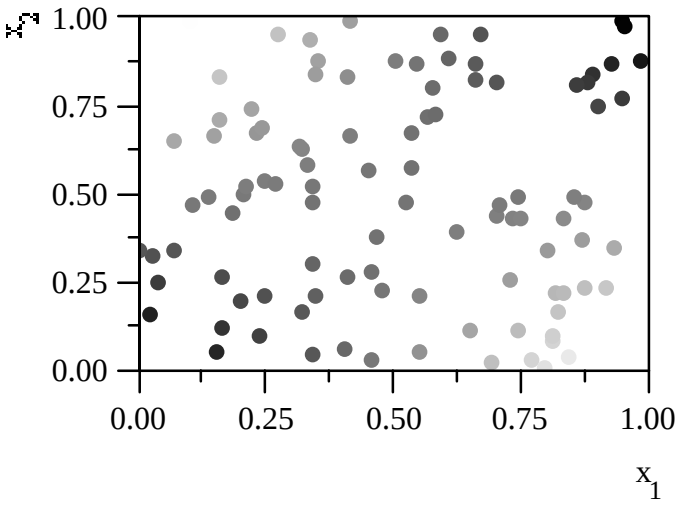


(a)

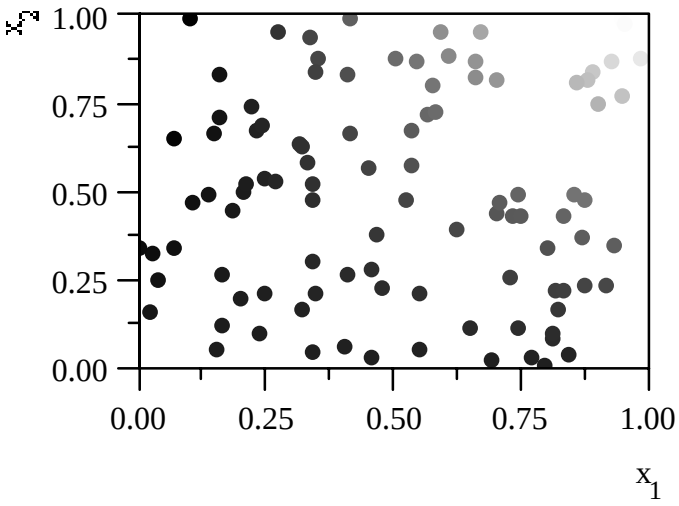


(b)

Figure 9. A set of training data (100 patterns): (a) first class, and (b) second class; the darker the pattern, the lower its class membership grade.



(a)



(b)

Figure 10. A set of testing data (100 patterns): (a) first class, and (b) second class; the darker the pattern, the lower its class membership grade.

The learning results summarized in the form of the performance index are provided in Table 1. They show the behavior of the fuzzy Petri net on the training set vis-à-vis the results obtained for the testing set. Several conclusions can be drawn from these results:

Table 1. Performance of the fuzzy Petri net (training and testing set) for various number of the transition nodes.

number of transition nodes	2	3	4	5	6	7	8
training set	2.367	0.140	0.079	0.069	0.070	0.050	0.062
testing set	2.155	0.121	0.069	0.067	0.053	0.043	0.073

- It becomes apparent that the number of transition nodes equal to 4 gives rise to a useful architecture that is not excessively large and still produces good classification results. Going toward a higher number of the transition nodes and eventually accepting an excessive size of the network does not yield a significant decrease in the values of the optimized performance index.

- There is an apparent jump in the performance of the network equipped with two transitions and the other versions of the net equipped with three or more transitions.

The following series of figures, Figures 11 to 13, illustrate more details dealing with the learning and performance of the fuzzy Petri net. This concerns a way in which the learning proceeds, visualizes the resulting firing levels of the transitions of the net, and illustrates the values of the classification errors reported for the individual patterns.

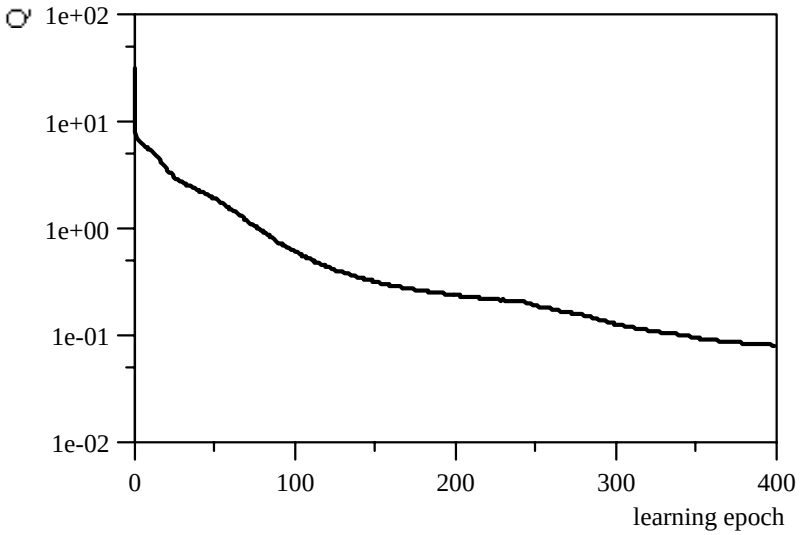
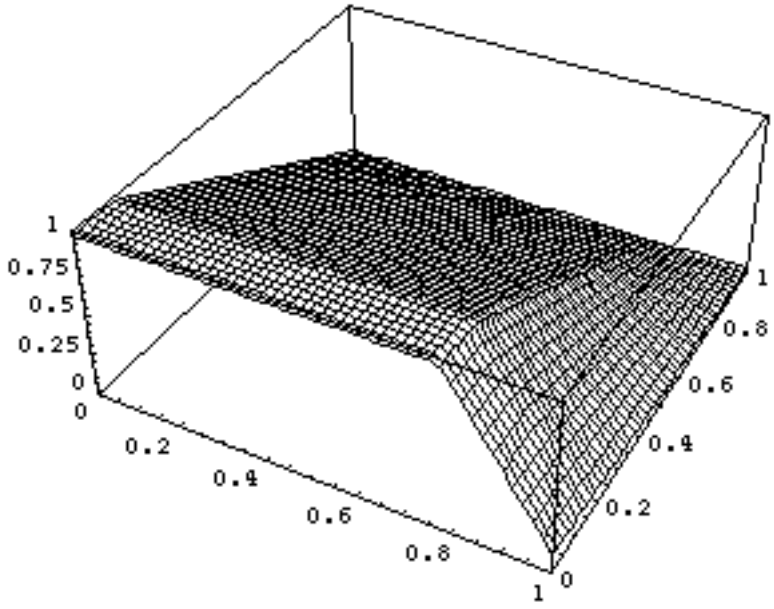
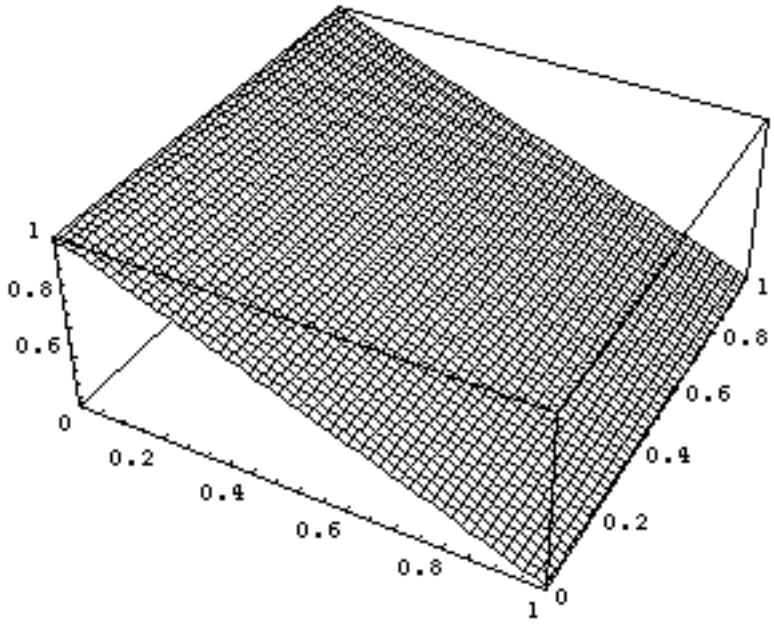


Figure 11. Performance index Q in successive learning epochs.

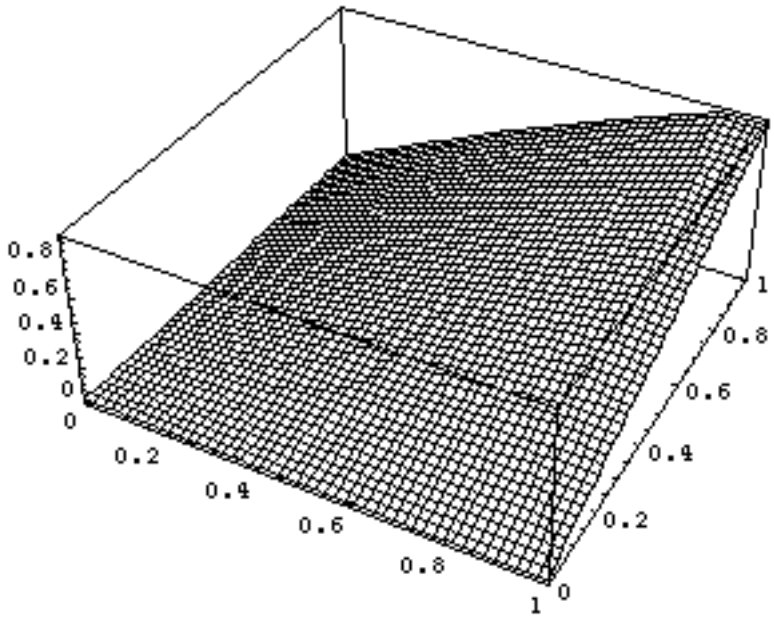


(a)

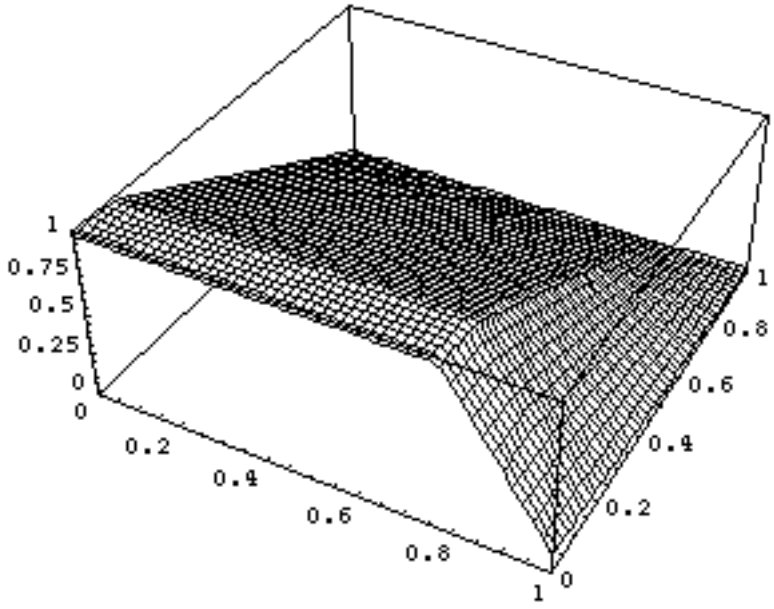


(b)

Figure 12. Characteristics of the transitions (transition nodes) regarded as functions of input variables x_1 and x_2 (continued on next page).

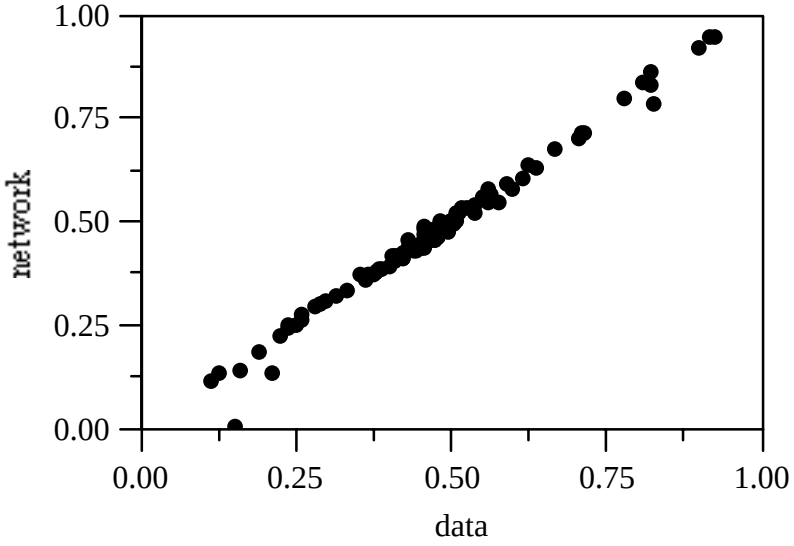


(c)

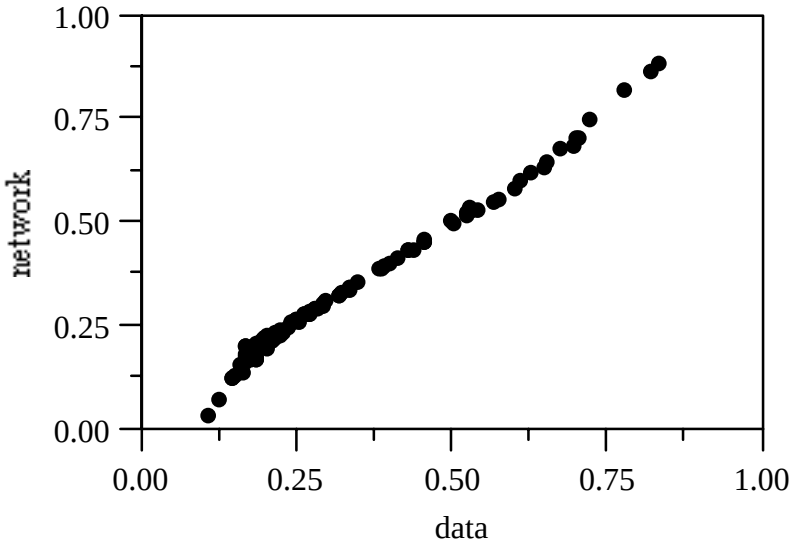


(d)

Figure 12 (continued).



(a)



(b)

Figure 13. Results of the network and the data: (a) first output; (b) second output.

The connections of the classifier are provided in the form of the following matrices (that are the connections and thresholds of the fuzzy Petri net):

$$\mathbf{W} = \begin{bmatrix} 0.9843 & 0.9968 & 0.0838 & 0.0066 \\ 0.9956 & 0.7385 & 0.4364 & 0.9986 \\ 0.0016 & 0.0332 & 0.9031 & 0.9950 \\ 0.0018 & 0.3866 & 0.6169 & 0.2966 \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} 0.0000 & 0.2674 & 0.2364 & 0.8721 \\ 0.5869 & 0.0000 & 0.9722 & 0.7987 \\ 0.9106 & 0.9763 & 0.6158 & 0.9749 \\ 0.7429 & 0.0000 & 0.2157 & 0.8697 \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} -3.8568 & 2.1462 & -4.3472 & 3.6591 \\ 0.6068 & -2.1980 & 3.2047 & -0.8616 \end{bmatrix}$$

Experiment 2. Here we study two other two-variable fuzzy functions governed by the expressions

$$f_1(x_1, x_2) = [(0.3 s x_1)(1 - x_2)] s [(1 - x_1) x_2]$$

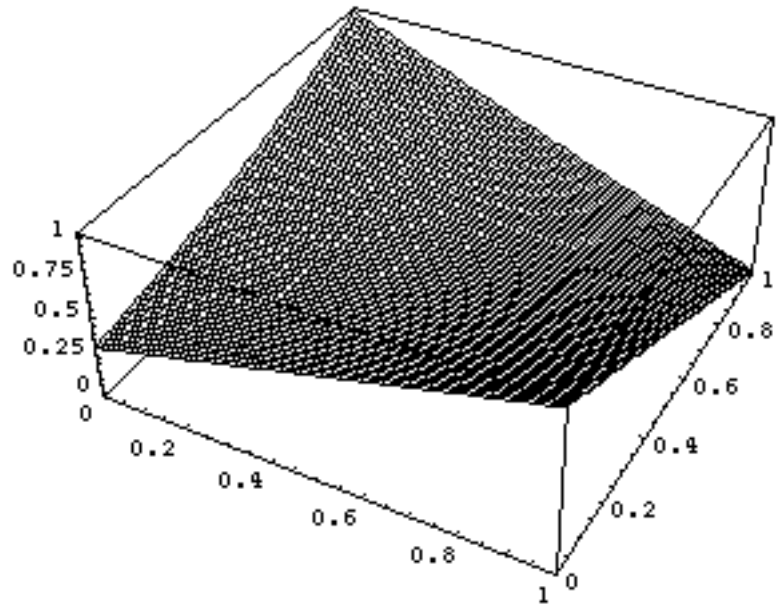
$$f_2(x_1, x_2) = [x_1 x_2] s [(1 - x_2)(0.4 s (1 - x_1))]$$

As a matter of fact, these give rise to the generalization of the exclusive-OR problem. The functions are also shown in Figure 14. The class boundaries clearly underline the nonlinear character of the classification problem, see Figure 15.

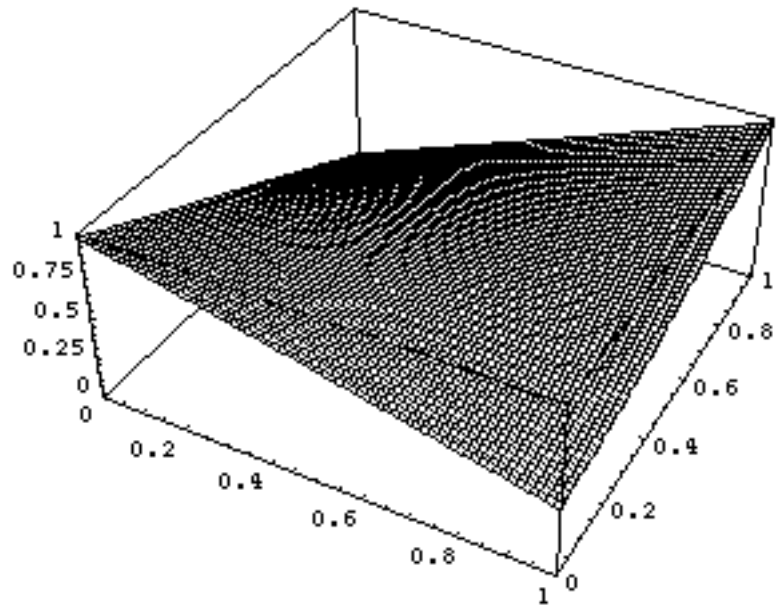
The results of learning for different sizes of the hidden layer (that is the number of transitions) are summarized in Table 2. Apparently the minimized error becomes significantly reduced at h = 5 and afterwards remains fairly stable (this effect is visible for both the learning and testing set).

Table 2. Performance of the fuzzy Petri net (both training and testing set) for various number of the transition nodes.

number of transition nodes	2	3	4	5	6	7	8
learning set	4.6357	0.5376	0.3087	0.0415	0.0424	0.0487	0.0463
testing set	6.0824	0.6922	0.4991	0.0986	0.1169	0.1057	0.0976



(a)



(b)

Figure 14. 3-D plots of the two-variable logic functions, f_1(a) and f_2(b).

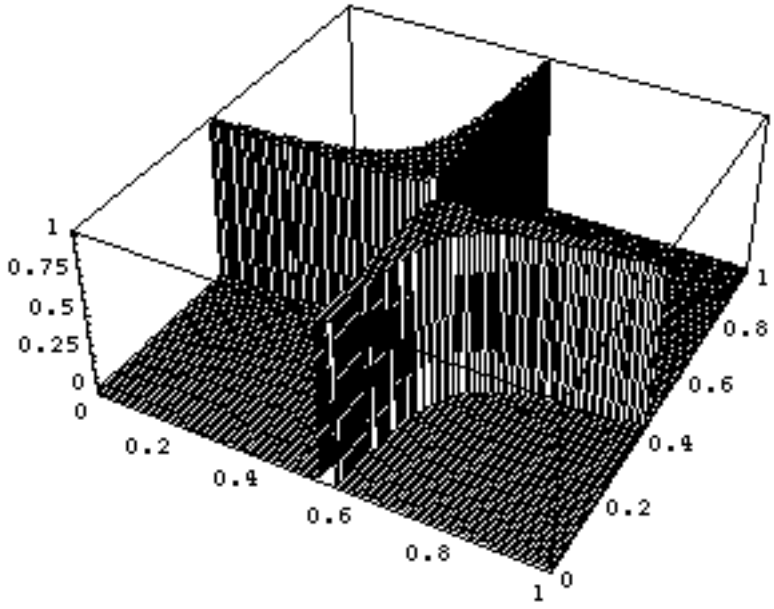


Figure 15. Classification boundaries in the two-class classification problem.

7 Conclusions

In this chapter, we have proposed a new approach to pattern classification, dwelling on the concept of the fuzzy Petri net. Two features of this architecture are definitely worth underlining. The first one concerns a transparent form of the classification model where each component of the fuzzy Petri net (places and transitions) comes as a clearly defined functional entity. The elements in the transition layer give rise to the combination of the original features thus producing new aggregates (synthetic features). The output places are used to aggregate evidence about class membership. Secondly, the Petri network exhibits a high level of parametric flexibility by coming equipped with a significant number of adjustable parameters (such as threshold levels of the transitions and the connections of the transition nodes as well as the output places).

The complete learning scheme has been proposed and illustrated with the aid of numeric examples. While the experiments dealt primarily with some specific forms of the t- and s-norms, it would be advisable to experiment with a wide range of such logic operators and view this as

an extra component of flexibility available in the design of such generalized Petri nets. The neuro-like style of performance of the proposed Petri net model being applied to classification problems provides us with a different and definitely interesting insight into the classification activities that is primarily based on features viewed as important resources utilized toward pattern classification.

The study has laid down the fundamentals of the new and general pattern recognition scheme. More specific application areas worth revisiting in this setting deal with scene analysis and computer vision where one can easily encounter parallel threads of classification pursuits.

Acknowledgment

Support from the Natural Sciences and Engineering Research Council of Canada (NSERC) is gratefully acknowledged.

References

[1] Cao, W.T. and Sanderson, A.C. (1995), "Task sequence planning using fuzzy Petri nets," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 25, pp. 755-768.

[2] Garg, M.L., Ahson, S.I., and Gupta, P.V. (1991), "A fuzzy Petri net for knowledge representation and reasoning," *Information Processing Letters*, vol. 39, pp. 165-171.

[3] Konar, A. and Mandal, A.K. (1996), "Uncertainty management in expert systems using fuzzy Petri nets," *IEEE Trans. on Knowledge and Data Engineering*, vol. 8, pp. 96-105.

[4] Looney, C.G. (1988), "Fuzzy Petri nets for rule-based decision making," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 18, pp. 178-183.

[5] Murata, T. (1989), "Petri nets: properties, analysis, and applications," *Proc. of the IEEE*, vol. 77, pp. 541-580.

[6] Pedrycz, W. and Gomide, F. (1994), “A generalized fuzzy Petri net model,” *IEEE Trans. on Fuzzy Systems*, vol. 2, pp. 295-301.

[7] Pedrycz, W. (1997), *Fuzzy Sets Engineering*, CRC Press, Boca Raton, Fl.