A Neuro Fuzzy Approach for Handling Structured Data

Alessio Ferone and Alfredo Petrosino

Department of Applied Science
University of Naples "Parthenope"
Centro Direzionale Isola C4
80143 Naples, Italy
{alessio.ferone,alfredo.petrosino}@uniparthenope.it

Abstract. Dealing with structured data has always represented a huge problem for classical neural methods. Although many efforts have been performed, they usually pre-process data and then use classic machine learning algorithm. Another problem that machine learning algorithm have to face is the intrinsic uncertainty of data, where in such situations classic algorithm do not have the means to handle them. In this work a novel neuro-fuzzy model for structured data is presented that exploits both neural and fuzzy methods. The proposed model called Fuzzy Graph Neural Network (F-GNN) is based on GNN, a model able to handle structure data. A proof of F-GNN approximation properties is provided together with a training algorithm.

Keywords: Structured pattern recognition, fuzzy systems, neural networks.

1 Introduction

Although neural methods have proved their powerful in dealing with various machine learning problems, sometimes they have failed because of the intrinsic data uncertainty. For this reason a growing interest has been addressed towards the integration of neural nets and approximated logics with a particular interested for the most famous, i.e. fuzzy logic[18,19]. Synergy between these two computational methods leads to the definition of new models called neuro-fuzzy systems. The basic idea behind neuro-fuzzy system is transforming fuzzy control system so to get neural net learning feature and hence exploiting advantages from both models. In neuro-fuzzy models, neural nets supply to fuzzy systems the connectionist structure and the learning ability, while fuzzy systems allow the use of a framework for approximated reasoning by means of rules in the IF-THEN form.

Fuzzy logic and neural nets are complementary technologies. Neural nets get information from systems that have to learn and control, while fuzzy logic based techniques use linguistic information from experts. It is natural that the synergy between these two techniques brings benefits to the final model. For example, it

© Springer-Verlag Berlin Heidelberg 2008

is possible to use a fuzzy system to represent experience from an expert and to use neural nets to calibrate their operation. In general, neuro-fuzzy model can be classified in three categories[6]:

1. *Neural fuzzy systems*: neural nets are used into fuzzy model
2. *Fuzzy neural networks*: fuzzification of neural nets
3. *Fuzzy-neural hybrid systems*: hybrid systems that incorporate fuzzy techniques and neural techniques.

In the first approach the goal is to provide fuzzy systems with tuning techniques used for neural nets without modifying their functionalities. In these models neural nets are used for numeric processing of fuzzy systems.

A fuzzy system of the first type are the *fuzzy basis function network* (FBFN), a type of neuro-fuzzy controller originally proposed by Wang and Mendel in [14]. In a FBFN, a fuzzy system is presented like an expansion series of fuzzy basis function (FBF), that is algebraic superimposition of membership functions. For this reason each FBF codes a fuzzy rule.

In the second approach, some typical elements of neural nets are fuzzified. In particular in fuzzy neural nets fuzzy neuron with activation signal obtained from a fuzzy relation are used[9].

In the third approach, both techniques play a fundamental role: each one operates in distinct part of the system so to incorporate complete functionalities of the other one[2].

A major drawback of the existing neural fuzzy systems is that their application domain is limited to static problems due to their inherent feedforward network structure. However, these kinds of studies are very interesting in all the application domains where the patterns are strongly correlated through structure, the processing is both numerical and symbolic and the nature of the data is imprecise and incomplete. Hence neural fuzzy systems capable for solving structure dependent problems are needeed.

The use of more complex data structures can lead to a better representation of data, so to simplify the solution of a given problem that deals with such data. Many efforts have been performed to handle structured data by pre-processing them and then apply classical machine learning algorithms. This approach not only add complexity to the final algorithm, but also introduce approximation errors and implementation difficulties. Moreover these kind of techniques tend to be specific for a problem and hence can be hardly reused for other problems. Neural methods are an example of techniques that evolved to handle with structured data[8][16][17][10]: original connectionist models have been modified to process sequences, trees and graphs. Some models[3][1] have been proposed to deal directly with structured data, also able to approximate showing in probability every function defined on graph till an arbitrary precision degree.

Recently, much work has been focused on the representational capabilities of recursive networks. The idea which motivates these studies is that if a network model cannot *represent* a certain structure, then it certainly cannot *learn* it either. The main question is then whether or not a given recursive network architecture can represent a specific structure [4]. Recursive neural networks can be initialized with prior knowledge provided by training structured data; this ability makes recursive neural networks useful tools for modeling tree automata, where prior knowledge is available.

The purpose of the present paper is to report how to construct neural fuzzy models for dealing with structured data in recursive manner. The work extends the model Graph neural network [11,12], letting to handle fuzziness. The learning algorithm of the reported model and its approximation capabilities are reported and demonstrated.

2 The Proposed F-GNN Model

The model is based on the idea that computing "concepts" is more efficient if performed by means of linguistic elements coded as fuzzy rules. A node can represent an object with some physical attributes, but it can also represent a concept; in this case a crisp computation is not adequate. A model able to elaborate such nodes should deal with an intrisic uncertainty, hence fuzzy logic can support this kind of computation with poweful tools. In the same way, edges represent relationships between nodes that can be better expressed in terms of linguistic concepts. Once again, fuzzy computation allows the use of powerful tools to deal with the uncertainty of these linguisitc elements. Based on these considerations, the information processing at each node and each edge is made by means of fuzzy systems in the form of *multi-input-single-output* (MISO). The idea is that each node computes input information using fuzzy rules.

Let us assume the fuzzy rules for each node have the following form:

$$R^j\text{: IF } x_1 \text{ is } A_1^j \text{ AND } x_2 \text{ is } A_2^j \text{ AND ... AND } x_n \text{ is } A_n^j \text{ THEN } y \text{ is } B^j$$

where $x_i, i = 1, 2, ..., n$ are the input variables, y is the output variable and A_i^j and B^j are linguistic terms characterized by membership functions $\mu_{A_i^j}(x_i)$ and $\mu_{B^j}(y)$ with $j = 1, 2, ..., M$.

Each node in the graph is encoded by a fuzzy control multi-input-single-output system ($X \subset \mathbb{R}^n \to Y \subset \mathbb{R}$) and in case of multiple output system, it can be decomposed in more multi-input-single-output systems.

Given an input graph, an encoding map is built using a MISO for each node of the graph. Each connection between nodes of the graph corresponds to connection between MISOs. Moreover for each output node another MISO is used. Figure 1 shows the structure of an encoding map where f_w and g_w are MISOs.

Connections between nodes are necessary to collect the states from adjacent nodes to a given node. In this way a node output will depend not only on the state and labels of the node itself but also on states and labels of adjacent nodes. Once the encoding map has been built, it is unfolded following the input graph connections (fig.2) and, once the fixed state has been reached, g-unit are added (encoding network). g-unit is unique in case of focused map; in general, we can say that they equal the number of output nodes.

From the definition of MISO and from the structure of encoding network, it is possible to note how in the proposed model f-unit input data are fuzzified and

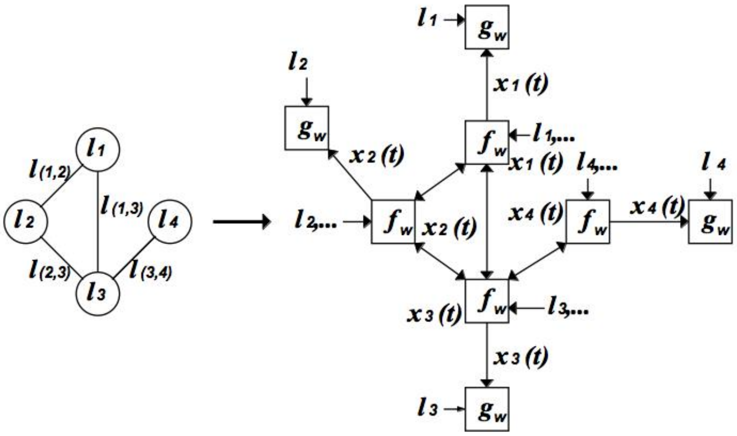

Fig. 1. Structure of an encoding map

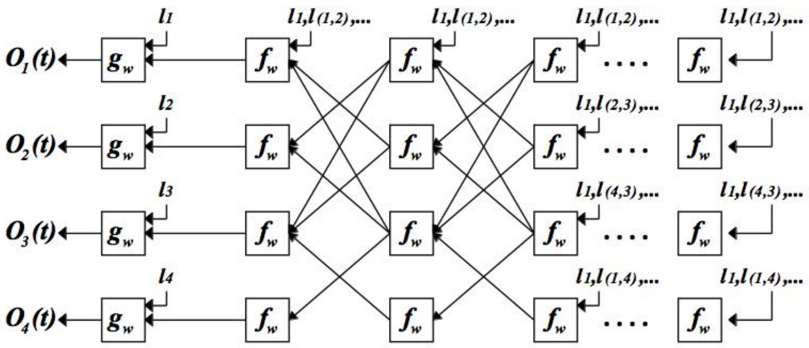

Fig. 2. Unfolding

defuzzified before they become input of the next f-unit. It is well known [5] that when more fuzzy rules are linked, problems can arise if the output of the first rule is not defuzzified before it is given in input to the next rule: this problem, called *fuzzy modus ponens*[13], has been faced by a large number of fuzzy logic researchers.

The above described procedure can be formalized by the following equation:

$$\begin{cases} x_n = f_w(l_n, l_{cp[n]}, x_{ne[n]}, l_{ne[n]}) \\ o_n = g_w(x_n, l_n) \end{cases} \tag{1}$$

where l_n is the label of node n, $l_{cp[n]}$ are the labels of its edges, $x_{ne[n]}$ are the states of the nodes in the neighborhood of n, $l_{ne[n]}$ are the labels of the nodes in the neighborhood of n, and x_n is the state of node n, i.e. it represents the concept denoted by node n. This value, along with the label l_n, is used to produce an output, i.e. a decision about the concept.

However, for non-positional graphs it is useful to replace function f_w of Eq.1 with the following equation

$$x_n = \sum_{u \in ne[n]} h_w(l_n, l_{(n,u)}, x_u, l_u) \tag{2}$$

where h_w is a parametric function. This transition function, already used in recursive neural networks, is not affected by the positions and the number of the children.

Assuming that the fuzzy system implemented on each node is composed by the following elements:

- Singleton fuzzifier
- Product inference
- Centroid defuzzifier
- Gaussian membership function.

Each MISO could be modeled as follows:

$$y = f(x) = \frac{\sum_{j=1}^{M} \overline{y}^j (\prod_{i=1}^{n} \mu_{A_i^j}(x_i))}{\sum_{j=1}^{M} (\prod_{i=1}^{n} \mu_{A_i^j}(x_i))} \tag{3}$$

where $f : X \subset \mathbb{R}^n \to \mathbb{R}$, $\overline{y}^j$ is the point in the output space Y where $\mu_{B^j}(\overline{y}^j)$ reaches its maximum value and $\mu_{A_i^j}(x_i)$ is a Gaussian membership function defined as:

$$\mu_{A_i^j}(x_i) = a_i^j exp[-\frac{1}{2}(\frac{x_i - m_i^j}{\sigma_i^j})^2] \tag{4}$$

where a_i^j, m_i^j and σ_i^j are real parameters and $0 < a_i^j \leq 1$.

From Eq. 4, it is possible to define *fuzzy basis functions* (FBFs) as:

$$p_j(\mathbf{x}) = \frac{\prod_{i=1}^{n} \mu_{A_i^j}(x_i)}{\sum_{j=1}^{m} (\prod_{i=1}^{n} \mu_{A_i^j}(x_i))}, j = 1, 2, ..., M. \tag{5}$$

and a *FBF network* (FBFN) like a FBF expansion:

$$f(\mathbf{x}) = \sum_{j=1}^{M} p_j(\mathbf{x}) \theta^j \tag{6}$$

where $\theta^j = \overline{y}^j \in \mathbb{R}$, i.e. a FBFN can be thought as a linear combination of FBFs.

This asserts also the computational equivalence between a MISO fuzzy system and a FBFN.

Figure 3 shows the structure of a FBFN where Gaussian membership functions composed with product are present in layer 1. The output of layer 1 is composed using the sums in layer 2, while in layer 3 the output is defuzzified. It has to be noted that, although FBFN structure for f-unit and g-unit is the same, rules implemented by the two nets are not necessarily the same.

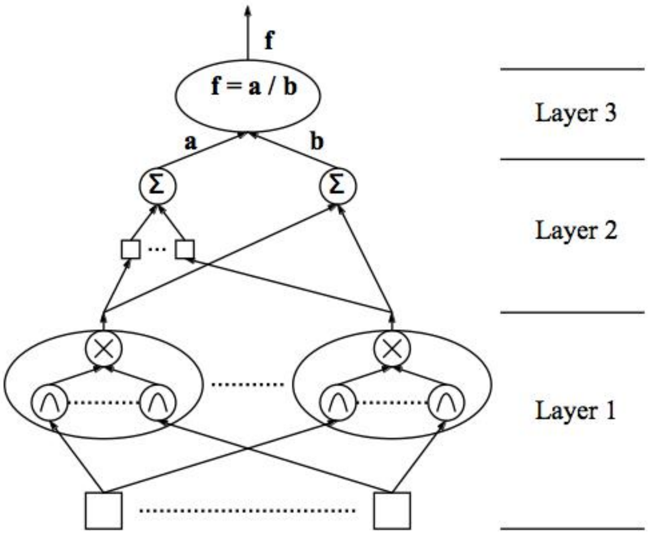

Fig. 3. FBFN structure

3 F-GNN: Universal Approximator

In the following, theorems that demonstrate approximation properties of the F-GNN model will be reported.

Firstly we introduce the concept of *unfolding equivalence*. Let us assume that $\tau : D \rightarrow R^m$ is a generic map constrained to produce the same output on nodes that are equivalent, i.e. given two nodes n and u of a grah G, $n \sim u$ implies $\tau(G, n) = \tau(G, u)$. The equivalence $\sim$ is called *unfolding equivalence* and is defined using the concept of *unfolding tree* T_n^d, that is the graph obtained by unfolding G up to the depth d using n as root.

Definition 1 Unfolding Equivalence. *Let $G = (N, E)$ be an undirected graph*[1]*. The nodes $n, u \in N$ are said to be unfolding equivalent, i.e. $n \sim u$, if $T_n = T_u$.*

Theorem 1 Approximation by non positional GNN*[11]. Let $\mathcal{D}$ be the non-positional graph domain. For each measurable function $\tau \in \mathcal{F}(\mathcal{D})$ that preserves the unfolding equivalence, each norm $\|\cdot\|$ over $\mathbb{R}^m$, each probability measure P over $\mathcal{D}$ and every real ϵ, μ, λ with* $\epsilon > 0, 0 < \lambda < 1, 0 < \mu < 1$, *exist two continuous derivable function h and g*

$$\begin{aligned} x_n &= \textstyle\sum_{u \in ne[n]} h_w(l_n, l_{(n,u)}, x_u, l_u) \\ o_n &= g_w(x_n, l_n), n \in N \end{aligned} \tag{7}$$

[1] The definition can be extended also to directed graph.

such that the global transition function F is a contraction with constant μ, state dimension is $s = 1$ and the function defined by $\phi(G, n) = o_n$ satisfy the condition

$$P(\|\tau(G, n) - \phi(G, n)\| \geq \epsilon) \leq 1 - \lambda \tag{8}$$

where the global transition function F is a stacked version of $|N|$ instances of f_w.

Theorem 2 Approximation by non-positional non-linear F-GNN. *Let assume true the hypothesis of the previous theorem, then exists a set of parameters w and two FBFNs h_w and g_w with continuous derivative such that the thesis of the previous theorem is verified.*

Previous theorem relies on the following Lemma, which demonstrates that FBFNs are universal approximators.

Lemma 1 Stone-Weierstrass theorem. *Let Z be a set of continuous real functions on a compact set U. If*

1. *Z is an algebra, that is the set Z is closed with respect to sum, product and scalar product operations*
2. *Z separates points in U, that is for each $x, y \in U$, $x \neq y$*

then Z uniform closure consists of all the continuous real functions on U, that is (Z, d_∞) is dense in $(G[U], d_\infty)$.

<u>Proof</u>: *Firstly, we show the proof that (Y, d_∞) is an algebra. Let $f_1, f_2 \in Y$ such that:*

$$f_1(x) = \frac{\sum_{j=1}^{K1}(\overline{z}1^j \prod_{i=1}^{n} \mu_{A1_i^j}(x_i))}{\sum_{j=1}^{K1}(\prod_{i=1}^{n} \mu_{A1_i^j}} \tag{9}$$

$$f_2(x) = \frac{\sum_{j=1}^{K2}(\overline{z}2^j \prod_{i=1}^{n} \mu_{A2_i^j}(x_i))}{\sum_{j=1}^{K2}(\prod_{i=1}^{n} \mu_{A2_i^j}} \tag{10}$$

from which we have:

$$f_1(x) + f_2(x) = \frac{\sum_{j1=1}^{K1}\sum_{j2=1}^{K2}(\overline{z}1^{j1} + \overline{z}2^{j2})(\prod_{i=1}^{n} \mu_{A1_i^{j1}}(x_i)\mu_{A2_i^{j2}}(x_i))}{\sum_{j1=1}^{K1}\sum_{j2=1}^{K2}(\prod_{i=1}^{n} \mu_{A1_i^{j1}}(x_i)\mu_{A2_i^{j2}}(x_i))} \tag{11}$$

Given that $\mu_{A1_i^{j1}}$ and $\mu_{A2_i^{j2}}$ are Gaussian membership functions, their product $\mu_{A1_i^{j1}}\mu_{A2_i^{j2}}$ is still a Gaussian function, hence $f_1 + f_2$ is in the form a FBFN ($f_1 + f_2 \in Y$). Similarly we have that:

$$f_1(x)f_2(x) = \frac{\sum_{j1=1}^{K1}\sum_{j2=1}^{K2}(\overline{z}1^{j1}\overline{z}2^{j2})(\prod_{i=1}^{n} \mu_{A1_i^{j1}}(x_i)\mu_{A2_i^{j2}}(x_i))}{\sum_{j1=1}^{K1}\sum_{j2=1}^{K2}(\prod_{i=1}^{n} \mu_{A1_i^{j1}}(x_i)\mu_{A2_i^{j2}}(x_i))} \tag{12}$$

is also in the form of a FBFN, hence $f_1 f_2 \in Y$. Finally, for an arbitrary $c \in \mathbb{R}$,

$$cf_1(x) = \frac{\sum_{j=1}^{K1}(c\overline{z}1^j \prod_{i=1}^{n} \mu_{A1_i^j}(x_i))}{\sum_{j=1}^{K1}(\prod_{i=1}^{n} \mu_{A1_i^j})} \tag{13}$$

we still have a FBFN, hence $f_1 f_2 \in Y$.

Next step consist in demonstrating that (Y, d_∞) *separates points* U.

Build a function $f \in Y$, *such that* $f(x_0) \neq f(y_0)$ *with arbitrary* $x_0, y_0 \in U$ *and* $x_0 \neq y_0$. *Then two fuzzy rules are chosen in the form:*

$$R_j\text{: IF } x_1 \text{ is } A_1^j \text{ AND } x_2 \text{ is } A_2^j \text{ ... AND } x_n \text{ is } A_n^j \text{ THEN } z \text{ is } B^j$$

as base fuzzy rule $(M = 2)$.

Let $x^0 = (x_1^0, x_2^0, ..., x_n^0)$ *and* $y^0 = (y_1^0, y_2^0, ..., y_n^0)$. *Se* $x_i^0 \neq y_i^0$, *two fuzzy sets are defined* $(A_i^1, \mu_{A_i^1})$ *and* $(A_i^2, \mu_{A_i^2})$ *where*

$$\mu_{A_i^1}(x_i) = exp[-\frac{(x_i - x_i^0)^2}{2}] \tag{14}$$

$$\mu_{A_i^2}(x_i) = exp[-\frac{(x_i - y_i^0)^2}{2}] \tag{15}$$

If $x_i^0 = y_i^0$, *then* $A_i^1 = A_i^2$ *and* $\mu_{A_i^1} = \mu_{A_i^2}$, *hence only a fuzzy set is defined. Two fuzzy sets are defined* $(B_i^1, \mu_{B_i^1})$ *and* $(B_i^2, \mu_{B_i^2})$ *where*

$$\mu_{B_i^j}(z) = exp[-\frac{(z - \overline{z}^j)^2}{2}] \tag{16}$$

with $j = 1, 2$ *and* $\overline{z}^j$ *specified in the following. Hence all parameters have been defined except* $\overline{z}^j$ $(j = 1, 2)$, *then a function* f, *in the FBFN form, has been defined, with* $M = 2$ *and* $\mu_{A_i^j}$ *specified before. With such* f *we have:*

$$f(x^0) = \frac{\overline{z}^1 + \overline{z}^2 \prod_{i=1}^n exp[-(x_i^0 - y_i^0)^2/2]}{1 + \prod_{i=1}^n exp[-(x_i^0 - y_i^0)^2/2]} = \alpha\overline{z}^1 + (1-\alpha)\overline{z}^2 \tag{17}$$

$$f(y^0) = \frac{\overline{z}^2 + \overline{z}^1 \prod_{i=1}^n exp[-(x_i^0 - y_i^0)^2/2]}{1 + \prod_{i=1}^n exp[-(x_i^0 - y_i^0)^2/2]} = \alpha\overline{z}^2 + (1-\alpha)\overline{z}^1 \tag{18}$$

where

$$\alpha = \frac{1}{1 + \prod_{i=1}^n exp[-(x_i^0 - y_i^0)^2/2]} \tag{19}$$

Given that $x^0 \neq y^0$, *it must exist some* i *such that* $x_i^0 \neq y_i^0$, *so that* $\prod_{i=1}^n exp[-(x_i^0 - y_i^0)^2/2] \neq 1$, *or* $\alpha \neq 1 - \alpha$. *If* $z^1 = 0$ *and* $z^2 = 1$, *then* $f(x^0) = 1 - \alpha \neq \alpha = f(y^0)$. *Hence* (Y, d_∞) *separates all points in* U.

(Y, d_∞) *in not null for any point in* U, *just choosing all the* $\overline{z}^j > 0$ *with* $j = 1, 2, .., M$, *i.e. every* $f \in Y$ *with* $\overline{z}^j > 0$ *can be used as the requested* f.

4 F-GNN: Training Algorithm

Before introducing F-GNN training algorithm, we have to show how a neuro-fuzzy system and a FBFN in particular can be trained:

1. If a_i^j, m_i^j and σ_i^j are free parameters, then the FBFN is non-linear and an optimization technique like backpropagation has to be used.
2. If all the parameters in $p_j(\mathbf{x})$ are given and the only free parameter is θ^j, then $f(\mathbf{x})$ is linear and the *orthogonal least-squares* (OLS)[14] algorithm can be used.

In particular in the second case, *Gram-Schmidt OLS* algorithm is employed, which can be determine automatically the number of significant FBFs. This algorithm is a hybrid learning method because other than the number of the most significant FBFs, it can compute the relative parameters.

4.1 FBFN Backpropagation Algorithm

In the most general case, where optimization has to be performed with respect to all the parameters, backpropagation algorithm is adopted for FBFN[15]. In particular FBFN has to be continuous differentiable with respect to parameters m, σ, θ, where m and σ represent mean and variance of the Gaussian membership function and θ represents the weight associated with each fuzzy basis function. Differentiability conditions are ensured by use of Gaussian membership function, because it is continuous differentiable with respect to both m and σ, hence FBFN is differentiable too with respect to the same parameters and also with respect to states x. For parameter θ, FBFN compact form

$$f(\mathbf{x}) = \sum_{j=1}^{M} p_j(\mathbf{x})\theta^j \tag{20}$$

shows that f is differentiable with respect to θ.

Given a pair $(\underline{x}^P, d^P)$, $\underline{x}^P \in U \subset \mathbb{R}^n$ and $d^P \in \mathbb{R}$, we want to minimize

$$e^P = \frac{1}{2}[f(\underline{x}^P) - d^P]^2 \tag{21}$$

where f is a FBFN. If we consider e, f, d as $e^P, f(\underline{x}^P), d^P$, respectively, we have:

1. To modify $\overline{z}^j$ it is used:

$$\overline{z}^j(k+1) = \overline{z}^j(k) - \alpha \frac{\partial e}{\partial \overline{z}^j} | \overline{z}^j = \overline{z}^j(k) \tag{22}$$

where $j = 1, ..., M$ and $k = 0, 1, 2, ...$ and α is the learning rate. Because f and hence e depend on $\overline{z}^j$ only through a, where $f = a/b$ with $a = \sum_{j=1}^{M}(\overline{z}^j y^j)$ and $b = \sum_{j=1}^{M} y^j$ and $y^j = \prod_{i=1}^{n} \mu_{A_i^j}(x_i^P)$, using the chain rule we have:

$$\frac{\partial e}{\partial \overline{z}^j} = (f - d)\frac{\partial f}{\partial a}\frac{\partial a}{\partial \overline{z}^j} = (f - d)\frac{1}{b}y^j \tag{23}$$

from which:

$$\overline{z}^j(k+1) = \overline{z}^j(k) - \alpha\frac{f - d}{b}y^j \tag{24}$$

where $j = 1, ..., M$ e $k = 0, 1, 2, ...$

2. To modify $\overline{x}_i^j$ it is used:

$$\overline{x}_i^j(k+1) = \overline{x}_i^j(k) - \alpha \frac{\partial e}{\partial \overline{x}_i^j} | \overline{x}_i^j = \overline{x}_i^j(k) \tag{25}$$

   where $i = 1, 2, ..., n$, $j = 1, 2, ..., M$ and $k = 0, 1, 2, ...$ Because f and hence e depend on $\overline{x}_i^j$ only through y^j, using the chain rule we have:

$$\frac{\partial e}{\partial \overline{x}_i^j} = (f - d)\frac{\partial f}{\partial y^j}\frac{\partial y^j}{\partial \overline{x}_i^j} = (f - d)\frac{\overline{z}^j - f}{b} y^j \frac{\overline{x}_i^P - \overline{x}_i^j}{\sigma_i^{j2}} \tag{26}$$

   from which:

$$\overline{x}_i^j(k+1) = \overline{x}_i^j(k) - \alpha \frac{f - d}{b}(\overline{z}^j - f) y^j \frac{\overline{x}_i^P - \overline{x}_i^j(k)}{\sigma_i^{j2}(k)} \tag{27}$$

   where $j = 1, ..., M$ e $k = 0, 1, 2, ...$
3. Using the same method, we have the following update of parameter σ_i^j:

$$\begin{aligned} \sigma_i^j(k+1) &= \sigma_i^j(k) - \alpha \frac{\partial e}{\partial \sigma_i^j} | \sigma_i^j = \sigma_i^j(k) \\ &= \sigma_i^j(k) - \alpha \frac{f - d}{b}(\overline{z}^j - f) y^j \frac{(x_i^P - \overline{x}_i^j(k))^2}{\sigma_i^{j3}(k)} \end{aligned} \tag{28}$$

   where $j = 1, ..., M$ and $k = 0, 1, 2, ...$

Based on such considerations, F-GNN learning algorithm see F as a stacked version of the FBFNs which implement f-units, G is the stacked version of the FBFNs which implement g-units and vector w contains free parameters to be modified (mean and variance of the membership functions and weights associated with each FBF).

To obtain a fast convergence of the algorithm, it is important to initialize parameters with values consistent with input data, rather than random. Mean and variance of the nodes label membership functions and of the edges labels membership functions can be computed as sampled mean and variance (or by using a clustering algorithm) because these data are part of the input data. On the contrary, parameters of the states membership functions cannot be computed

Algorithm 1. F-GNN Training algorithm

```
1: x = FORWARD-1ST(w)
2: repeat
3:     $\frac{\partial e_w}{\partial w}$=BACKWARD(x,w)
4:     w = w - $\lambda \cdot \frac{\partial e_w}{\partial w}$
5:     x = FORWARD(w);
6: until stop criterion is reached
7: return w
```

previously and hence it is necessary, during operation of FORWARD-1st function, to apply a clustering algorithm or to compute sampled mean and variance of the produced states.

Next, the output node states are first computed (by FORWARD step) and then computes the gradient (by BACKWARD step).

Once all parameters have been updated, a new FORWARD step is performed, until the desired accuracy is reached.

5 Conclusion

The aim of the proposed research is the theoretic definition of a neuro-fuzzy model for structured data. The model employs neural nets to capture information contained in each node of a graph as well as the information contained at each adjacent node. This allows to produce a new state, that is a concept defined by a node and its adjacent nodes. Motivations behind the study of the proposed model relies on the consideration that a concept can be expressed more effectively in terms of natural language rather than with numeric values. This aspect imposes the use of a logic that could handle linguistic terms in the framework of a numeric model. The chosen logic is the one that more then the others has contributed to bring "human experience" into expert systems: fuzzy logic.

Although fuzzy logic is often used in static system, the challenge for many researchers has been that of introducing it into dynamic systems, like neural nets, which has lead to a new field of computer science: neural fuzzy systems. This is particularly demanded when the data are too complex, incomplete and, as the topic of the present paper, are characterized by an inner structure. Synergy of systems based on fuzzy logic and neural nets to deal structured data aims to put together two complementary techniques, so to get the best from both of them. This synergy is mainly required in many application fields where structured data play a fundamental role, like bioinformatics and computer vision.

The reported model is in this direction and extends in the fuzzy framework recent works on this subject. In the reported model, named by us F-GNN, nodes and edges labels (which represents an event) are fuzzy data as well as fuzzy results the state computation. A remarkable result is presented for the proposed model in that it is a universal approximator. A learning algorithm based on the FBFN backpropagation is also sketched to handle structured data.

References

1. Bianchini, M., Maggini, M., Sarti, L., Scarselli, F.: Recursive neural networks for processing graphs with labelled edges: Theory and applications. Neural Networks - Special Issue on Neural Network and Kernel Methods for Structured Domains 18, 1040–1050 (2005)
2. Challoo, R., Clark, D.A., McLauchlan, R.A., Omar, S.I.: A fuzzy neural hybrid system. IEEE World Congress on Computational Intelligence 3, 1654–1657 (1994)

3. Gori, M., Maggini, M., Sarti, L.: A Recursive neural network model for processing directed acyclic graph with labeled edges. In: Procedings of the International Joint Conference on Neural Networks, vol. 2, pp. 1351–1355 (2003)
4. Gori, M., Petrosino, A.: Encoding nondeterministic fuzzy tree automata into recursive neural networks. IEEE Transactions on Neural Networks 15(6), 1435–1449 (2004)
5. Gorrini, V., Bersini, H.: Recurrent fuzzy systems. FUZZ-IEEE World Congress on Computional Intelligence, pp. 193–198 (1994)
6. Lin, C.T., Lee, C.S.G.: Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems, p. 482. Prentice-Hall, Englewood Cliffs (1996)
7. Lin, C.T., Lee, C.S.G.: Real-time supervised structure/parameter learning for fuzzy neural networks. In: Proc. IEEE International Conference of Fuzzy Systems, vol. 2(1), pp. 1283–1291 (1992)
8. Nowlan, S.J.: Gain variation in recurrent error propagation networks. Complex Systems 2, 305–320 (1988)
9. Pal, S.K., Mitra, S.: Multilayer perceptron, fuzzy sets, and classification. IEEE Trans. Neural Networks 3(5), 683–697 (1992)
10. Pineda, F.: Generalization of back-propagation to recurrent neural networks. Physical Review Letters 59, 2229–2232 (1987)
11. Scarselli, F., Gori, M., Tsoi, A.-C., Hagenbuchner, M., Monfardini, G.: The Graph Neural Network Model. IEEE Transactions on Neural Networks (to appear, 2008)
12. Scarselli, F., Gori, M., Tsoi, A.-C., Hagenbuchner, M., Monfardini, G.: Computational Capabilities of Graph Neural Networks. IEEE Transactions on Neural Networks (to appear, 2008)
13. Smets, P.: Implications and Modus Ponens in Fuzzy Logic. In: Conditional Logic in Expert Systems, pp. 235–268. Elsevier Science Publisher, Amsterdam (1991)
14. Wang, L.X., Mendel, J.M.: Fuzzy basis functions, universal approximation, and orthogonal least-squres learning. IEEE Transaction on Neural Networks 3(5), 807–814 (1992)
15. Wang, L.X., Mendel, J.M.: Back-Propagation fuzzy system as nonlinear dynamic system identifier. IEEE Transaction on Neural Networks 3(5), 1409–1418 (1992)
16. Werbos, P.J.: Backpropagation through time: what it does and how to do it. Proceedings of the IEEE 78(10), 1550–1560 (1990)
17. Williams, R., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. Neural Computation (1), 270–280 (1989)
18. Zadeh, L.: Fuzzy sets. Information and Control 8(3), 338–353 (1965)
19. Zadeh, L.: Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems 1, 3–28 (1978)