Assessment of EFuNN Accuracy for Pattern Recognition Using Data with Different Statistical Distributions

Ronei Marcos de Moraes

Department of Statistics. Federal University of Paraíba - Cidade Universitária s/n - 58051-900 - João Pessoa/PB - Brazil
{ronei@de.ufpb.br}

Abstract. This work assesses the accuracy of Evolving Fuzzy Neural Networks (EFuNNs) for pattern recognition tasks using seven different statistical distributions data. The recently proposed EFuNNs are dynamic connectionist feed forward networks with five layers of neurons and they are adaptive rule-based systems. Results of assessment are provided and show different accuracy according to the statistical distribution of data.

Keywords: Evolving Fuzzy Neural Networks, Pattern Recognition, Accuracy Assessment.

1 Introduction

In 2001 Kasabov [5] proposed a new class of Fuzzy Neural Networks named Evolving Fuzzy Neural Networks (EFuNNs). EFuNNs are structures that evolve according determined principles. EFuNNs have low complexity and high accuracy which are important features to a pattern recognition method. In several papers EFuNN showed best results when compared to Multilayer Perceptron Neural Networks (MLP) [4, 7, 8, 9]. However, it is known that results of both nets depend on the training data for a specific applications. In particular, results are dependent on statistical distribution of training data. Previous works related to analysis of results on EFuNN with respect the statistical distributions of data were not found.

In this paper we made an assessment of accuracy of EFuNNs in pattern recognition tasks using seven different statistical distributions, with 1, 2, 3 and 4 dimensions for each. Results of those comparisons are provided with an analysis about the better kind of statistical distribution of data to be used for better EFuNN performance for each dimension of data.
The paper is organized as following: in the Section 2 is presented theoretical aspects of Evolving Fuzzy Neural Networks. Section 3 brings details about assessment criteria, statistical distributions used and Kappa Coefficient. Section 4 presents an analysis of results and Section 5 shows the conclusions of the comparisons.

2 Evolving Fuzzy Neural Networks

As mentioned before, Evolving Fuzzy Neural Networks (EFuNNs) are structures that evolve according ECOS principles [3]: quick learning, open structure for new features and new knowledge, representing space and time and analyze itself of errors. The EFuNN is a connectionist feed forward network with five layers of neurons, but nodes and connections are created or connected when data examples are presented [5]. The input layer represents input variable of the network as crisp value \( x \). The second layer represents fuzzy quantization of inputs variables. Here, each neuron implements a fuzzy set and its membership function (MF) as triangular membership, Gaussian membership or other. The third layer contains rule nodes \( r_j \) that evolve through training. Each one is defined by two connections vectors: \( W_1(r_j) \) from fuzzy input layer to rule nodes and \( W_2(r_j) \) from rule nodes to fuzzy output layer. These nodes are created during network learning and they represent prototypes of data mapping from fuzzy input to fuzzy output space. In the third layer we can use a linear activation function or a Gaussian function. The fourth layer represents fuzzy quantization of the output variables from a function of inputs and from an activation function. The last layer uses an activation function to calculate defuzzified values for output variables \( y \).

In the third layer, each \( W_1(r_j) \) represents the coordinates of the center of a hypersphere in the fuzzy input space and each \( W_2(r_j) \) represents the coordinates of the center of a hypersphere in the fuzzy output space. The radius of the hypersphere of a rule node \( r_j \) is defined as \( R_j = 1 - S_j \), where \( S_j \) is the sensitive threshold parameter for activation of \( r_j \) from a new example \((x,y)\). The pair of fuzzy data \((x_f, y_f)\) will be allocated to \( r_j \) if \( x_f \) is into the \( r_j \) input hypersphere and if \( y_f \) is into the \( r_j \) output hypersphere. For this, two conditions must be satisfied:

a) The local normalized fuzzy distance between \( x_f \) and \( W_1(r_j) \) must be smaller than \( R_j \).

The local normalized fuzzy distance between these two fuzzy membership vectors is done by:

\[
D( x_f, W_1(r_j) ) = \| x_f - W_1(r_j) \| / \| x_f + W_1(r_j) \| \quad (1)
\]

where \( \| a - b \| \) and \( \| a + b \| \) are the sum of all the absolute values of a vector that is obtained after vector subtraction \( a - b \) or summation \( a + b \) respectively.
b) The normalized output error $Err = \frac{|| y - y' ||}{N_{out}}$ must be smaller than an error threshold $E$, where $y$ is as defined before, $y'$ is produced by EFuNN output, $N_{out}$ is the number of outputs and $E$ is the error tolerance of the system for fuzzy output.

$$W_1(r_j^{(t+1)}) = W_1(r_j^{(t)}) + l_{j,1}[W_1(r_j^{(t)}) - x_f]$$

$$W_2(r_j^{(t+1)}) = W_2(r_j^{(t)}) + l_{j,2}(A_2 - y_f)A_1(r_j^{(t)})$$

where $l_{j,1}$ is the learning rate for the first layer and $l_{j,2}$ is the learning rate for the second layer. In general, it can be assumed they have the same value done by: $l_j = 1/N_{ex}(r_j)$, where $N_{ex}(r_j)$ is the number of examples associated with rule node $r_j$.

$$A_1(r_j^{(t)}) = f_1\left(D(W_1(r_j^{(t)})), x_f\right)$$

is the activation function of the rule $r_j^{(t)}$ and

$$A_2 = f_2( W_2 A_1 )$$

is the activation of the fuzzy output neurons, when $x$ is presented. For the functions $f_1$ and $f_2$ a simple linear function can be used.

When a new example is associated with a rule $r_j$, the parameters $R_j$ and $S_j$ are changed:

$$R_j^{(t+1)} = R_j^{(t)} + D\left(W_1(r_j^{(t+1)}), W_1(r_j^{(t)})\right)$$

$$S_j^{(t+1)} = S_j^{(t)} - D\left(W_1(r_j^{(t+1)}), W_1(r_j^{(t)})\right)$$

If exists temporal dependencies between consecutive data, the connection weight $W_3$ can capture that. The connection $W_3$ works as a Short-Term Memory and as a feedback connection from rule nodes layer. If the winning rule node at time $(t - 1)$ was $r_{max}^{(t-1)}$ and at time $(t)$ was $r_{max}^{(t)}$, then a link between the two nodes is established by [6]:

$$W_3\left[r_{max}^{(t-1)}, r_{max}^{(t)}\right]^{new} = W_3\left[r_{max}^{(t-1)}, r_{max}^{(t)}\right]^{old} + l_3 A_1(r_{max}^{(t-1)}) A_1(r_{max}^{(t)})$$

where $A_1(r_{max}^{(t)})$ denotes the activation of a rule node $r$ at a time $(t)$ and $l_3$ defines a learning rate. If $l_3 = 0$, then no temporal associations are learned in an EFuNN.

The EFuNN learning algorithm starts with initial values for parameters [5]. According to mentioned above, the EFuNN is trained by examples until convergence. When a new data example $d = (x, y)$ is presented, the EFuNN either creates a new rule $r_n$ to memorize the new data (input vector $W_1(r_n) = x$ and output vector $W_2(r_n) = y$) or adjusts the winning rule node $r_j$ [5].
3 Assessment

In pattern recognition tasks is necessary to use a methodology that can provide better performance over any kind of data. Unfortunately, several methods proposed in the literature can obtain their better performance when specific statistical distribution of data is used. Some methods can obtain good results with different kind of distributions, as neural networks. As a neural network, an EFuNN is a structure that evolves, but in some papers better results were obtained when its use is compared with a neural network in a same application [4, 8, 9].

Based on that results, we investigated in this paper the behavior of EFuNN for pattern recognition tasks, according on kinds of statistical distributions of data utilized. We made simulations with seven different statistical distributions: Binomial, Continuous Uniform, Discrete Uniform, Exponential, Gaussian, Poisson and Weibull [2]. For each statistical distribution were analyzed four different dimensions. Our goal is to know for what statistical distribution we can use EFuNN and what performance is expected for that statistical distribution. Besides that, we can know for what statistical distribution of data the better performance of EFuNN can be expected.

In this paper we made an assessment of accuracy of EFuNNs in pattern recognition tasks using seven different statistical distributions, with 1, 2, 3 and 4 dimensions for each. Results of those comparations are provided and also an analysis about the better kind of statistical distribution of data to be used for better EFuNN performance for each dimension of data.

For performance assessment, we use the matrix of classification and Kappa Coefficient [1]. The Kappa Coefficient was proposed by Cohen in 1960 and is used to compare classification performance of a classifier, using a reference classification and a classification matrix. Kappa Coefficient is presented bellow:

\[
P_0 = \frac{\left( \sum_{i=1,M} n_{ii} \right)}{N};
\]

\[
P_c = \frac{\left( \sum_{i=1,M} n_{i+} n_{+i} \right)}{N^2};
\]

\[
K = \frac{P_0 - P_c}{(1 - P_c)}
\]

where \( n_{ii} \) is the sum of line \( i \) of classification matrix and \( n_{i+} \) is the sum of column \( i \) of the same matrix. The variance \( \sigma^2_K \) is done by:

\[
\theta_1 = \frac{\left[ \sum_{i=1,M} n_{ii} (n_{i+} + n_{+i}) \right]}{N^2};
\]

\[
\theta_2 = \frac{\left[ \sum_{i=1,M} n_{ii} (n_{i+} + n_{+i})^2 \right]}{N^3};
\]
\[ \theta_1 = \frac{P_0 (1 - P_0)}{N (1 - P_c)^2}; \quad (12) \]

\[ \theta_2 = \frac{2 (1 - P_0) + 2 P_0 P_c - \theta_1}{N (1 - P_c)^3}; \quad (13) \]

\[ \theta_3 = \frac{(1 - P_0)^2 \theta_2 - 4 P_c^2}{N (1 - P_c)^4}; \quad (14) \]

\[ \sigma_K^2 = \theta_1 + \theta_2 + \theta_3. \quad (15) \]

For calculations, it can be used an approximation to the first component of equation (15). However, in this paper we use the complete formula for variance.

4 Results

In this section we presented the main results of tests in pattern recognition tasks using EFuNN. Only the better results are presented due to space restrictions. We generate two sets of samples: the first with 500 samples for training and the second with 4000 samples for classification test. Each sample had four different classes, with four dimensions for each one of seven statistical distributions. All the results presented below are about samples for classification test.

In all Figures below, we are presenting sample data for training an EFuNN in the first line. The next line shows the sample data for testing with the same parameters. For sample data, we generated 1500 samples, but the first 1000 was discarded. For sample data for testing, we generated 5000 samples, but the first 1000 was discarded. The first two lines in the Figure correspond to the first dimension (sample of train and sample for testing). The next two for the second dimension and following in that sequence. It is possible to observe for some classes the great intersection between the distributions. This has been done intentionally to make simulations were as similar as possible to real conditions.

4.1 Binomial Distribution

The Binomial distribution can model a discrete random variable with only two types of occurrence. For one dimension, the parameters used were: number of membership functions (MF) 6, no-prunning, sensitivity threshold value 0.999 and error threshold value 0.001. With that configuration, Mean Square Error was 0.2116, the percentile of correct classification was 78.8%, with 848 classification mistakes and Kappa Coefficient was 71.7333%, with variance $6.6945 \times 10^{-5}$. When dimension of data was changed for 2, the parameters used were the same, except the MF, which was changed for 5. The Mean Square Error was 0.1347, the percentile of correct classification was 86.875%, with 525 classification mistakes and Kappa Coefficient was 82.5000%,
with variance $4.9923 \times 10^{-5}$. For dimension 3, all parameters remained the same and all points were classified correctly, i.e. Kappa Coefficient was 100%, with variance zero. For dimension 4, the same was observed. From that results, we can observe that EFuNN is able to classify data from Binomial distribution with good accuracy. For dimensions over three it can obtain 100% of correct classification.

![Fig. 1. Random numbers generated for Binomial distribution.](image)

### 4.2 Continuous Uniform Distribution

The Continuous Uniform Distribution is used when all possible results have the same probability of occurrence in continuous space. The parameters used for one dimension were: MF 6, no-pruning, sensitivity threshold value 0.999 and error threshold value 0.001. From that configuration, Mean Square Error obtained was 0.0645, the percentile of correct classification was 93.575%, with 257 classification mistakes and Kappa Coefficient was 91.4333%, with variance $2.6721 \times 10^{-5}$. For dimension 2, the parameters used were the same, except the MF, changed for 2. In this case, all points were classified correctly (Kappa Coefficient was 100% and variance was zero). The same was observed for dimensions 3 and 4. The EFuNN seems to be adapted to classify data with this distribution. It is capable to classify data with relative accuracy even in low dimensions and when classes intersection happens.
Fig. 2. Random numbers generated for Continuous Uniform distribution.

Fig. 3. Random numbers generated for Discrete Uniform distribution.
4.3 Discrete Uniform Distribution

The Discrete Uniform Distribution is used when all possible results have the same probability of occurrence in discrete space. The parameters used for one dimension were: MF 4, no-pruning, sensitivity threshold value 0.999 and error threshold value 0.001. Using that configuration, the Mean Square Error obtained was 0.0164, the percentile of correct classification was 98.35%, with 66 classification mistakes and Kappa Coefficient was 97.8000%, with variance $7.2072 \times 10^{-6}$. For dimension 2, the parameters used were the same and all points were classified correctly. The same was observed for dimensions 3 and 4. So, the EFuNN also seems to be adapted to classify data with this distribution. It is capable to classify data with relative accuracy, even in low dimensions and when classes intersection happens. The same verification done for the continuous uniform distribution can be done for the discrete uniform distribution.

4.4 Exponential Distribution

The Exponential Distribution calculates the time elapsed between two events of interest. For one dimension, the parameters used were: number of membership functions (MF) 9, no-pruning, sensitivity threshold value 0.999 and error threshold value 0.001. With that configuration, Mean Square Error was 1.3110, the percentile of correct classification was 40.025%, with 2399 classification mistakes and Kappa Coefficient was 20.0333%, with variance $1.1082 \times 10^{-4}$. When dimension of data was changed for 2, the parameters used were the same, except the MF, which was changed for 5. The Mean Square Error was 1.1751, the percentile of correct classification was 46.025%, with 2159 classification mistakes and Kappa Coefficient was 28.0333%, with variance $1.1191 \times 10^{-4}$. For dimension 3, all parameters remained the same and the Mean Square Error was 1.0161, the percentile of correct classification was 47.15%, with 2114 classification mistakes and Kappa Coefficient was 29.5333%, with variance $1.1105 \times 10^{-4}$. For dimension 4, the parameters used were the same, except the MF, which was changed for 3. The Mean Square Error was 0.9584, the percentile of correct classification was 52.0250%, with 1919 classification mistakes and Kappa Coefficient was 52.0250%, with variance $1.1056 \times 10^{-5}$. In opposition to the uniform distribution, the EFuNN is not suitable to classify data from exponential distribution and it did not obtain good results for none of the researched dimensions.
Fig. 4. Random numbers generated for Exponential distribution.

Fig. 5. Random numbers generated for Gaussian distribution.
4.5 Gaussian Distribution

The Gaussian or Normal Distribution is the most typical distribution and it is used to model several independent variables as the social-economic variables. The parameters used for one dimension were: MF 7, no-prunning, sensitivity threshold value 0.999 and error threshold value 0.001. From that configuration, Mean Square Error obtained was 0.2642, the percentile of correct classification was 72.70005%, with 1092 classification mistakes and Kappa Coefficient was 63.6000%, with variance $8.6015 \times 10^{-5}$. For dimension 2, the parameters used were the same and the Mean Square Error was 0.2007, the percentile of correct classification was 80.1750%, with 793 classification mistakes and Kappa Coefficient was 73.5667%, with variance $7.0074 \times 10^{-5}$. In the case of dimension 3, the Mean Square Error was 0.1296, the percentile of correct classification was 88.9000%, with 444 classification mistakes and Kappa Coefficient was 85.2000%, with variance $4.3852 \times 10^{-5}$. For dimension 4, the parameters used were the same, except the MF, which was changed for 6. The Mean Square Error was 0.0484, the percentile of correct classification was 97.2250%, with 111 classification mistakes and Kappa Coefficient was 96.3000%, with variance $1.1987 \times 10^{-5}$. We can observe that EFuNN obtained good results, however it did not obtain total correct classification for any of the researched dimensions.

4.6 Poisson Distribution

The Poisson Distribution is used to model discrete events in continuous space or time. The parameters used for one dimension were: MF 3, no-prunning, sensitivity threshold value 0.999 and error threshold value 0.001. Using that configuration, the Mean Square Error obtained was 0.040, the percentile of correct classification was 94.6750%, with 213 classification mistakes and Kappa Coefficient was 92.9000%, with variance $2.2283 \times 10^{-5}$. For dimension 2, the parameters used were the same, except the MF, which was changed for 2, and all points were correctly classified (Kappa Coefficient was 100% and variance was zero). The same was observed for dimensions 3 and 4. The EFuNN seems to be adapted to classify data from Poisson distribution better than any other researched. Even with dimension 1 it obtained a high rate of correct classification.
4.7 Weibull Distribution

The Weibull Distribution is used to model rates of displacement in equipments and their durability. The parameters used for one dimension were: MF 5, no-pruning, sensitivity threshold value 0.999 and error threshold value 0.001. From that configuration, Mean Square Error obtained was 1.5351, the percentile of correct classification was 37.3000%, with 2508 classification mistakes and Kappa Coefficient was 16.4000%, with variance $1.0455 \times 10^{-4}$. For dimension 2, the parameters used were the same, except the MF, changed for 2. In this case, the Mean Square Error obtained was 1.0363, the percentile of correct classification was 47.4000%, with 2104 classification mistakes and Kappa Coefficient was 29.8667%, with variance $1.1141 \times 10^{-4}$. For dimension 3, the parameters used were the same, except the MF, which was changed again for 6. In this case, the Mean Square Error obtained was 0.4516, the percentile of correct classification was 73.4250%, with 1063 classification mistakes and Kappa Coefficient was 64.5667%, with variance $8.6754 \times 10^{-5}$. Again, the MF parameter was altered for the dimension 4, taking on the value 3. For that dimension, the Mean Square Error obtained was 0.2362, the percentile of correct classification was 90.7750%, with 369 classification mistakes and Kappa Coefficient was 87.7000%, with variance $3.7221 \times 10^{-5}$. We can observe that EFuNN obtained good results, however it did not obtain total correct classification for any of the researched dimensions.

Fig. 6. Random numbers generated for Poisson distribution.
Fig. 7. Random numbers generated for Weibull distribution.

The Table 1 presents a summary of best results obtained by each statistical distribution in this simulation. For distributions Binomial, Discrete and Continuous Uniform and Poisson distributions, using EFuNN is possible to achieve 100% in pattern recognition tasks, according to the Kappa Coefficient. However, for Exponential distribution, its performance is modest. For data which follows Gaussian or Weibull distributions, EFuNN is a competitive approach and the results are better according to higher dimensions of data.

Table 1. Summary of bests by statistical distributions, according to the Kappa Coefficient.

<table>
<thead>
<tr>
<th>Statistical Distribution</th>
<th>Number of Dimensions</th>
<th>Kappa Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binomial</td>
<td>3 or more</td>
<td>100.000 %</td>
</tr>
<tr>
<td>Continuous Uniform</td>
<td>2 or more</td>
<td>100.000 %</td>
</tr>
<tr>
<td>Discrete Uniform</td>
<td>2 or more</td>
<td>100.000 %</td>
</tr>
<tr>
<td>Exponential</td>
<td>4</td>
<td>52.025 %</td>
</tr>
<tr>
<td>Gaussian</td>
<td>4</td>
<td>96.300 %</td>
</tr>
<tr>
<td>Poisson</td>
<td>2 or more</td>
<td>100.000 %</td>
</tr>
<tr>
<td>Weibull</td>
<td>4</td>
<td>87.700 %</td>
</tr>
</tbody>
</table>
5 Conclusions and Further Works

In this paper we presented an assessment of Evolving Fuzzy Neural Networks (EFuNNs) accuracy for pattern recognition using data with different statistical distributions. We made simulations with seven different statistical distributions: Binomial, Continuous Uniform, Discrete Uniform, Exponential, Gaussian, Poisson and Weibull. For each statistical distribution were analyzed four different dimensions according to Mean Square Error, percentile of correct classification, number of classification mistakes, Kappa Coefficient and its variance.

According to the results obtained, EFuNN could be recommended to classify data from the Binomial, Discrete and Continuous Uniform and Poisson distributions. In opposition, it seems to be not suitable the use of EFuNN to classify data from the Exponential distribution. For the Gaussian and Weibull distributions, EFuNN is able to perform a good classification, but the increasing of accuracy is related to higher dimensions of data.

As future work, we intend to expand the studies to data dimensions higher than those studied in this work. Another possibility to be explored is combining different distributions for a same simulation.

Acknowledgments. This work is partially supported by grant 310339/2009-0 of the Brazilian Council for Scientific and Technological Development, CNPq. This work is related to the National Institute of Science and Technology for Medicine Assisted by Scientific Computing also supported by CNPq.

References

Assessment of EFuNN Accuracy for Pattern Recognition Using Data with Different Statistical Distributions