Fuzzy-Based Machine Learning Algorithm for Intelligent Systems

K Pradheep Kumar

Abstract It has become essential to develop machine learning techniques due to the automation of various tasks. At present, several tasks need manual intervention for better reliability of the system. In this work, fuzzy-based approach has been proposed where systems are trained based on initial data sets. In several data sets, the data is either partially available or unavailable. When data sets need to be used on real time systems, the non-availability of data may lead to catastrophe. In this approach, a fuzzy-based rule set is formulated. The rule strength is used to determine the effectiveness. Rules with similar strengths are clustered together. The learning is carried out by determining a threshold for the formulated rule set. Based on the threshold computed, a modified rule set is formulated with rule strengths greater than the computed threshold. A semi-supervised learning approach that uses an activation function is employed. The fuzzy learning approach proposed in this work reduces the error by 20% compared to conventional approaches.

Keywords Threshold · Activation function · Learning rule matrix
Machine learning · Rule strength · Smart systems · Semi-supervised learning

1 Introduction

In today's world, there is a need to design gadgets and systems with automation. To enable automation of gadgets, it is essential that systems handle decisions independently based on the environmental conditions.

Systems should adapt to the external environment and handle tasks based on the underlying behaviour. Systems should also handle exceptional tasks. To fulfil the same, the system needs to be intelligent to take decisions in the absence of manual intervention.

K. Pradheep Kumar (✉)
BITS Pilani, Pilani, Rajasthan, India
e-mail: pradjourn@gmail.com

V. E. Balas et al. (eds.), *Data Management, Analytics and Innovation*,
Advances in Intelligent Systems and Computing 808,
https://doi.org/10.1007/978-981-13-1402-5_25

Machine learning is one way of implementing these features of Artificial Intelligence where systems are trained based on an initial data set.

The system forms the initial dataset by collecting the information from the external environment through sensors and actuators. System is trained to learn information and take decisions based on iteration of datasets. The time needed by the system to learn information and take decisions from the datasets should be finite and small so as to avoid failures that can lead to catastrophe.

In this work, a fuzzy rule matrix is constructed and the threshold of the entire rule set is also computed. The modified rule set is constructed by selecting rules whose rule strength is greater than the threshold. These are the rules used by the system to learn the data. Other rules are not considered.

The paper is organised as follows: Sect. 2 discusses the literature on machine learning algorithms. Section 3 explains the fuzzy model proposed, illustrates the same with an example and gives the algorithm. Section 4 explains the simulation environment and the results. Section 5 concludes the work and highlights the scope for the future work.

The block diagram of the entire procedure is shown in Fig. 1.

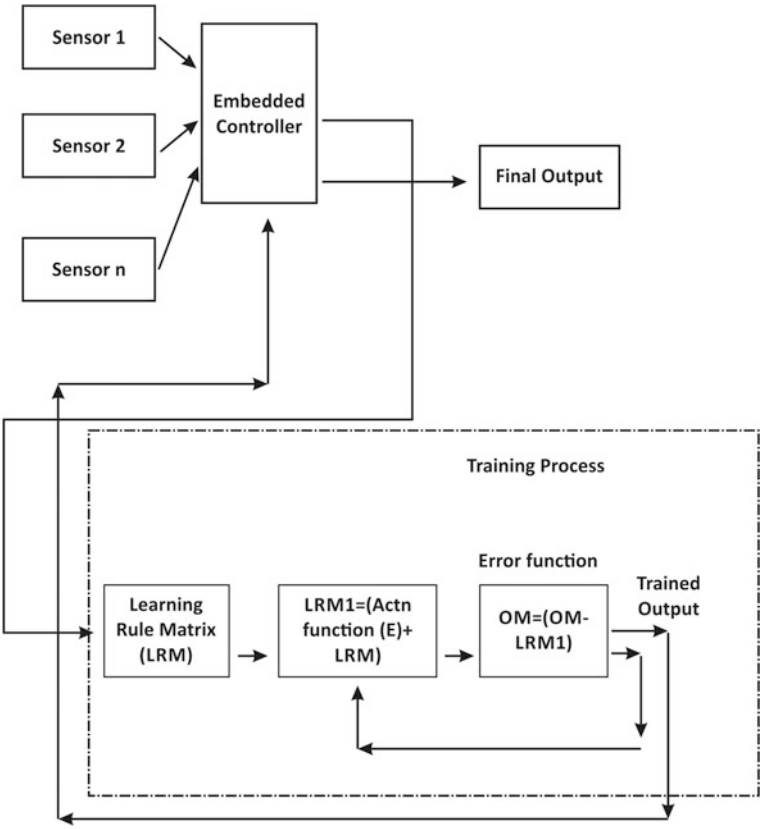

Fig. 1 Block diagram of fuzzy learning model

2 Literature Review

The most conventional machine learning strategy is the Naive Bayes classifier. Several literatures report the inaccuracies in Bayes' conditional independence. This is due to the suboptimal nature of the probability estimates as reported in [1–3]. Even when multidimensional tables are used to compute probabilities in real-time scenarios, many errors are observed as discussed in [2]. Support Vector method uses a hyperplane to classify the data. Based on the classification a number of support vectors are generated as explained in [4]. But when data sets are widely spread and have a large range of deviation, this method does not yield accurate results. The choice of the hyperplane is the key factor associated with the data classification as discussed in [5, 6]. Artificial neural networks give reliability but again this depends on the choice of the activation function used to train the data set as explained in [7, 8]. Linear and Logistic regression as explained in [9–11] uses a cost function and estimates the gradient. The reduction in the gradient is used to learn the datasets. The regression analysis depends on the effectiveness of the cost function.

Several deep learning neural network algorithms model a data set as a function using a mathematical model and train the same to interpolate or extrapolate the data value for a particular value. The Stochastic Gradient Descent technique as explained in [12] attempts to minimise the loss of the model by incorporating additional parameters and increments these proportional to the gradient estimated. But this makes the data noisy due and filters needs to be used to eliminate the noise associated. The method of steepest descent as discussed on [13] which when used on a very large data set requires an infinite number of iterations to get an optimum solution. It may so happen that an optimum solution may not exist also. Another approach which works in a different dimension is Reinforcement learning as explained in [14]. This method arrives at an optimal solution by trial and error in infinite time. But the time of training in an unsupervised scenario would be infinite and the optimal solution cannot be guaranteed. The drawbacks highlighted in the above techniques are overcome in the fuzzy-based analysis. In the fuzzy-based analysis a precise value for the rule strength are arrived and the optimal solution is arrived by training the same with an activation function in a finite period of time. It does not require any mathematical model and infinite number of trials to arrive at an optimal solution.

3 Proposed Work

In this work a fuzzy-based learning approach using Mamdani's engine as discussed by Venkat and Pradheep in [15] has been used. For each rule, a rule strength is computed. The rule strength is computed based on a set of sub-factors and their corresponding weights. For each linguistic variable used in the rule, the mid-value

of the range is computed. The weighted average of each variable of the rule gives the rule strength. A typical rule is given as follows:

The Rule Strength is computed using the expression

$$\mathrm{RS} = \frac{\sum (\mathrm{Weight} * \mathrm{MidValue})}{\sum \mathrm{Weight}} \tag{1}$$

The rule strengths have been computed for all rules in the fuzzy rule set using the above expression. The threshold of the fuzzy rule set is computed using the following expression

$$T = \frac{\sum_{i=1}^{i=n} (\mathrm{Rulenumber} * \mathrm{RuleStrength})}{\sum_{i=1}^{n} \mathrm{Rulenumber}} \tag{2}$$

where n is the total number of rules.

The threshold computed is a measure of the upper bound till completion of the training of the system. The modified rule set is constructed by selecting rules, which have a rule strength greater than the computed threshold. After this a matrix is constructed in the form (RS − mT, RS, RS + mT), where m = 1 to n and n is the number of rules in the original rule set. The matrix is formulated for different values of m in such a manner that RS − T is positive. This criterion decides the number of columns of the matrix.

The matrix now is non-symmetrical as the number of rows would be greater than the number of columns. The matrix is split into a number of sub-symmetrical matrices by matching the number of rows with the columns.

The procedure has been illustrated with an example where RS − T is positive and RS − 2T, RS − 3T, etc. are negative. With only RS − T, RS and RS + T, the matrix has been formulated. Hence, the modified rule set should be split into 3 X 3 matrices based on the modified rule set. The actual output matrix is accepted as input. The error, which is the difference between the output and modified rule matrix, is computed. An activation function is used to iteratively train the modified rule set till the error is non-negative. The final value is the optimal rule set.

For computing the trained matrix for antivirus check, the different sub-factors and their corresponding weights are shown in Table 1.

The different Linguistic variables with their ranges and corresponding mid-values are indicated in Table 2.

A typical rule is shown below.

If (Phishing Check is Excellent) and (Spyware Check is Good) and (Malware check is Average) and (Trojan Check is Fair) and (Rootkit scan is Poor) Then (Antivirus software is Average).

The entire rule set comprises of 125 rules. Few sample rules from the entire rule set is shown below in Table 3.

Table 1 Weight for each sub-factor

S.No.	Sub-factor	Weight
1.	Phishing check	5
2.	Spyware check	4
3.	Malware check	3
4.	Trojan check	2
5.	Rootkit scan	1

Table 2 Linguistic variable with range and mid-value

S.No.	Linguistic variable	Range	Mid-value
1.	Excellent	0–19	9.5
2.	Good	20–39	29.5
3.	Average	40–59	49.5
4.	Fair	60–79	69.5
5.	Poor	80–100	90

The rule strength of the rule If (Phishing Check is Excellent) and (Spyware Check is Good) and (Malware check is Average) and (Trojan Check is Fair) and (Rootkit scan is Poor), then (Antivirus software is Average) is 36.2. The Rule strengths of the above rule base would be computed as indicated in Table 4.

Using the expression, the threshold has been computed as 62.02. Based on this threshold modified rule set is shown in Table 5.

Hence RS − T is only non-negative and the rest of RS − mT is negative. From the above discussion 14 (3 X 3 Symmetrical) sub-matrices have been formulated. The last row of the 14th matrix would be zeroes. The modified rule set is trained using the activation function $E = \left(\frac{1}{1-e^{-T}}\right)$ as shown in Table 6.

The modified rule matrix is LRM. The iteration matrix is LRM1. Given Output matrix is OM. The error is (OM − LRM1). The iteration is carried out till all elements of OM − LRM1 becomes negative.

The algorithm has been discussed below

- Form the fuzzy rule set
- Compute the rule strength and Threshold
- Form the Learning rule matrix (LRM) with selected rules (RS > T)
- Form modified rule matrix (RS − mT, RS, RS + mT)
- Accept the output matrix (OM)
- Start the iteration by computing Error E = (OM − LRM)
- Use activation function $E = \left(\frac{1}{1-e^{-T}}\right)$ to minimise the error.
- LRM1 = LRM + E
- If (LRM1 <= OM) then
    - Final output O1 = LRM1
      Else
    - L1: LRM1 = LRM + E

Table 3 Sample rule set

S.No.	Malware check	Spyware check	Rootkit check	Trojan check	Phishing check	Antivirus software
1.	Excellent	Excellent	Excellent	Excellent	Excellent	Excellent
2.	Good	Good	Good	Good	Excellent	Good
3.	Average	Average	Average	Average	Excellent	Average
4.	Fair	Fair	Fair	Fair	Excellent	Fair
5.	Poor	Poor	Poor	Poor	Excellent	Poor
6.	Excellent	Excellent	Excellent	Excellent	Good	Excellent
7.	Good	Good	Good	Good	Good	Good
8.	Average	Average	Average	Average	Good	Average
9.	Fair	Fair	Fair	Fair	Good	Fair
10.	Poor	Poor	Poor	Poor	Good	Poor
11.	Excellent	Excellent	Excellent	Excellent	Average	Excellent
12.	Good	Good	Good	Good	Average	Good
13.	Average	Average	Average	Average	Average	Average
14.	Fair	Fair	Fair	Fair	Average	Fair
15.	Poor	Poor	Poor	Poor	Average	Poor
16.	Excellent	Excellent	Excellent	Excellent	Fair	Excellent
17.	Good	Good	Good	Good	Fair	Good
18.	Average	Average	Average	Average	Fair	Average
19.	Fair	Fair	Fair	Fair	Fair	Fair
20.	Poor	Poor	Poor	Poor	Fair	Poor
21.	Excellent	Excellent	Excellent	Excellent	Poor	Excellent
22.	Good	Good	Good	Good	Poor	Good
23.	Average	Average	Average	Average	Poor	Average
24.	Fair	Fair	Fair	Fair	Poor	Fair
25.	Poor	Poor	Poor	Poor	Poor	Poor
26.	Excellent	Excellent	Excellent	Excellent	Excellent	Excellent
27.	Good	Good	Good	Excellent	Good	Good
28.	Average	Average	Average	Excellent	Average	Average
29.	Fair	Fair	Fair	Excellent	Fair	Fair
30.	Poor	Poor	Poor	Excellent	Poor	Poor
31.	Excellent	Excellent	Excellent	Good	Excellent	Excellent
32.	Good	Good	Good	Good	Good	Good
33.	Average	Average	Average	Good	Average	Average
34.	Fair	Fair	Fair	Good	Fair	Fair
35.	Poor	Poor	Poor	Good	Poor	Poor

Table 4 Rule strengths computed from rule set

Rule no.	Malware check	Spyware check	Rootkit check	Trojan check	Phishing check	Rule strength = ((5 * phishing check) + (4 * spyware check) + (3 * malware check) + (2 * trojan check) + (1 * rootkit check))/15	Threshold parameter Q = (Rule no. * rule strength)
1	9.5	9.5	9.5	9.5	9.5	9.50	9.5
2	9.5	9.5	49.5	9.5	9.5	12.17	24.33
3	9.5	9.5	9.5	9.5	9.5	9.50	28.50
4	29.5	9.5	9.5	9.5	9.5	13.50	54.00
5	9.5	9.5	9.5	9.5	9.5	9.50	47.50
6	9.5	29.5	9.5	9.5	9.5	14.83	89.00
7	9.5	9.5	9.5	9.5	9.5	9.50	66.50
8	9.5	9.5	9.5	9.5	9.5	9.5	76.00
9	9.5	9.5	9.5	29.5	9.5	12.17	109.50
10	49.5	9.5	9.5	9.5	9.5	17.50	175.00
11	9.5	9.5	69.5	9.5	9.5	13.50	148.50
12	9.5	9.5	9.5	49.5	9.5	14.83	178.00
13	9.5	9.5	29.5	9.5	9.5	10.83	140.83
14	9.5	9.5	9.5	69.5	9.5	17.50	245.00
15	9.5	29.5	29.5	29.5	29.5	25.50	382.50
16	29.5	29.5	29.5	29.5	29.5	29.50	472.00
17	9.5	9.5	90	9.5	9.5	14.87	252.73
18	29.5	29.5	69.5	29.5	29.5	32.17	579.00
19	9.5	9.5	9.5	9.5	29.5	16.17	307.17
20	9.5	69.5	9.5	9.5	9.5	25.50	510.00
21	29.5	49.5	29.5	29.5	29.5	34.83	731.50
22	29.5	29.5	29.5	29.5	49.5	36.17	795.67

(continued)

Table 4 (continued)

Rule no.	Malware check	Spyware check	Rootkit check	Trojan check	Phishing check	Rule strength = ((5 * phishing check) + (4 * spyware check) + (3 * malware check) + (2 * trojan check) + (1 * rootkit check))/15	Threshold parameter Q = (Rule no. * rule strength)
23	29.5	29.5	29.5	29.5	29.5	29.50	678.50
24	9.5	49.5	9.5	9.5	9.5	20.17	484.00
25	9.5	9.5	9.5	90	9.5	20.23	505.83
26	29.5	29.5	29.5	29.5	69.5	42.83	1113.67
27	29.5	29.5	29.5	49.5	29.5	32.17	868.50
28	69.5	9.5	9.5	9.5	9.5	21.50	602.00
29	49.5	29.5	49.5	49.5	49.5	44.17	1280.83
30	29.5	29.5	29.5	29.5	29.5	29.50	885.00
31	29.5	29.5	29.5	29.5	9.5	22.83	707.83
32	9.5	9.5	9.5	9.5	49.5	22.83	730.67
33	29.5	29.5	29.5	69.5	29.5	34.83	1149.50
34	49.5	49.5	9.5	49.5	49.5	46.83	1592.33
35	29.5	29.5	29.5	29.5	29.5	29.50	1032.50

Table 5 Modified rule set

Modified rule no.	Malware check	Spyware check	Rootkit check	Trojan check	Phishing check	Rule strength (RS) = ((5 * phishing check) + (4 * spyware check) + (3 * malware check) + (2 * trojan check) + (1 * rootkit check))/15	(RS − T)	(RS + T)
1	69.50	49.50	69.50	69.50	69.50	64.17	2.15	126.19
2	49.50	69.50	69.50	69.50	69.50	65.50	3.48	127.52
3	69.5	69.5	29.5	69.5	69.5	66.83	4.81	128.85
4	69.5	69.5	69.5	69.5	69.5	69.50	7.48	131.52
5	69.5	69.5	69.5	29.5	69.5	64.17	2.15	126.19
6	69.5	69.5	9.5	69.5	69.5	65.50	3.48	127.52
7	69.5	69.5	69.5	49.5	69.5	66.83	4.81	128.85
8	90	90	90	90	90	90.00	27.98	152.02
9	69.5	69.5	69.5	69.5	69.5	69.50	7.48	131.52
10	90	90	90	90	90	90.00	27.98	152.02
11	69.5	69.5	69.5	69.5	49.5	62.83	0.81	124.85
12	49.5	49.5	49.5	49.5	90	63.00	0.98	125.02
13	90	90	90	90	9.5	63.17	1.15	125.19
14	90	90	90	90	90	90.00	27.98	152.02
15	69.5	69.5	49.5	69.5	69.5	68.17	6.15	130.19
16	90	9.5	90	90	90	68.53	6.51	130.55
17	90	90	90	90	29.5	69.83	7.81	131.85
18	69.5	69.5	90	69.5	69.5	70.87	8.85	132.89
19	90	90	90	90	90	90.00	27.98	152.02
20	69.5	69.5	69.5	90	69.5	72.23	10.21	134.25
21	90	69.5	69.5	69.5	69.5	73.60	11.58	135.62
22	90	29.5	90	90	90	73.87	11.85	135.89
23	9.5	90	90	90	90	73.90	11.88	135.92

Table 6 Trained modified rule set

LRM				LRM1			OM			OM − LRM1		
Rule	RS − T	RS	RS + T	RS − T	RS	RS + T	RS − T	RS	RS + T	RS – T	RS	RS + T
1	2.15	64.17	126.19	62.66	124.68	186.70	5.67	72.34	132.65	−56.99	−52.34	−54.05
2	3.48	65.50	127.52	63.99	126.01	188.03	10.67	77.34	137.65	−53.32	−48.67	−50.38
3	4.81	66.83	128.85	65.32	127.34	189.36	15.67	82.34	142.65	−49.65	−45.00	−46.71
4	7.48	69.50	131.52	67.99	130.01	192.03	20.67	87.34	147.65	-47.32	-42.67	-44.38
5	2.15	64.17	126.19	62.66	124.68	186.70	25.67	92.34	152.65	−36.99	−32.34	−34.05
6	3.48	65.50	127.52	63.99	126.01	188.03	30.67	97.34	157.65	−33.32	−28.67	−30.38
7	4.81	66.83	128.85	65.32	127.34	189.36	35.67	102.34	162.65	−29.65	−25.00	−26.71
8	27.98	90.00	152.02	88.49	150.51	212.53	40.67	107.34	167.65	−47.82	−43.17	−44.88
9	7.48	69.50	131.52	67.99	130.01	192.03	45.67	112.34	172.65	−22.32	−17.67	−19.38
10	27.98	90.00	152.02	88.49	150.51	212.53	50.67	117.34	177.65	−37.82	−33.17	−34.88
11	0.81	62.83	124.85	61.32	123.34	185.36	55.67	122.34	182.65	−5.65	−1.00	−2.71
12	0.98	63.00	125.02	61.49	123.51	185.53	60.67	127.34	187.65	−0.82	3.83	2.12
13	1.15	63.17	125.19	61.66	123.68	185.70	65.67	132.34	192.65	4.01	8.66	6.95
14	27.98	90.00	152.02	88.49	150.51	212.53	70.67	137.34	197.65	−17.82	−13.17	−14.88
15	6.15	68.17	130.19	66.66	128.68	190.70	75.67	142.34	202.65	9.01	13.66	11.95
16	6.51	68.53	130.55	67.02	129.04	191.06	80.67	147.34	207.65	13.65	18.30	16.59
17	7.81	69.83	131.85	68.32	130.34	192.36	85.67	152.34	212.65	17.35	22.00	20.29
18	8.85	70.87	132.89	69.36	131.38	193.40	90.67	157.34	217.65	21.31	25.96	24.25
19	27.98	90.00	152.02	88.49	150.51	212.53	95.67	162.34	222.65	7.18	11.83	10.12
20	10.21	72.23	134.25	70.72	132.74	194.76	100.67	167.34	227.65	29.95	34.60	32.89
21	11.58	73.60	135.62	72.09	134.11	196.13	105.67	172.34	232.65	33.58	38.23	36.52
22	11.85	73.87	135.89	72.36	134.38	196.40	110.67	177.34	237.65	38.31	42.96	41.25
23	11.88	73.90	135.92	72.39	134.41	196.43	115.67	182.34	242.65	43.28	47.93	46.22

- If (LRM1 > OM)
  - O1 = LRM1
    Else
  - (LRM1 = LRM1 + E)
- Repeat iteration until LRM1 value is > OM.
- Goto L1.
- Compute RMSE, RRSE, RAE and MAE for final value of O1.
- End the procedure.

4 Simulation Results

The algorithm was simulated on Java platform with several IT-related issues like Antivirus, Data backup strategies, hardware maintenance. Simulations were carried out on fuzzy rule sets with about 500 rules.

- Root Mean Square Error (RMSE)
- Root Relative Square Error (RRSE)
- Relative Absolute Error (RAE)
- Mean Absolute Error (MAE).

4.1 Root Mean Square Error (RMSE)

It is the square root of the mean of the squared difference between the Output Matrix (OM) and Trained Learning Rule Matrix (LRM1). It is given by the equation

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{i=n}(\mathrm{OM}(i) - \mathrm{LRM1}(i))^2} \tag{3}$$

where n is the number of sample values of the data set under consideration.

4.2 Root Relative Square Error (RRSE)

It is defined as the normalised version of the total squared error to the total squared error of the prediction made.

$$\text{RRSE} = \sqrt{\frac{\sum_{i=1}^{i=n} (\text{LRM1}(i) - \text{OM}(i))^2}{\sum_{i=1}^{i=n} (\text{OM}(i) - A\sim)^2}} \tag{4}$$

where $A\sim$ is the average of the n sample values.

$$A\sim = \frac{1}{n}\sum_{i=1}^{i=n} \text{OM}(i) \tag{5}$$

4.3 Relative Absolute Error (RAE)

It is defined as the total absolute error normalised to the total absolute error of the prediction made.

$$RAE = \frac{\sum_{i=1}^{i=n} |(LRM1(i) - OM(i))|}{\sum_{i=1}^{i=n} |(OM(i) - A\sim)|} \tag{6}$$

where A $\sim$ is the average of the n sample values.

$$A\sim = \frac{1}{n}\sum_{i=1}^{i=n} OM(i) \tag{7}$$

4.4 Mean Absolute Error (MAE)

It is defined as the error difference for the n samples considered.

$$\text{MAE} = \frac{\sum_{i=1}^{i=n} (\text{OM}(i) - \text{LRM1}(i))}{n} \tag{8}$$

The results of the fuzzy-based approach proposed in this work have been compared with the four conventional machine learning algorithms listed below:

1. Naïve Bayes Classifier
2. Support Vector Method (SVM)

3. Regression Analysis
4. Artificial Neural Networks (ANN).

4.5 Comparison with Naïve Bayes Classifier

The results of the proposed fuzzy-based approach are compared with Naïve Bayes Classifier. The reduction in Error for the parameters RMSE, RRSE, RAE and MAE when compared with the Naives Classifier method are 21, 17, 18 and 27% respectively as indicated in Table 7.

4.6 Comparison with Support Vector Method (SVM)

The reduction in Error for the parameters RMSE, RRSE, RAE and MAE for the proposed work when compared with the Support Vector Method are 32, 10, 17 and 17% respectively as indicated in Table 8.

4.7 Comparison with Regression Analysis

The reduction in Error for the parameters RMSE, RRSE, RAE and MAE for the proposed fuzzy-based approach when compared with the Regression Analysis Method are 13, 4, 7 and 23% respectively as indicated in Table 9.

4.8 Comparison with Artificial Neural Networks (ANN)

The reduction in Error for the parameters RMSE, RRSE, RAE and MAE for the proposed fuzzy-based approach when compared with the Artificial Neural Networks method are 34, 21, 17 and 32% respectively as indicated in Table 10.

4.9 Summary

The reduction in error for the proposed work compared to conventional approaches is shown in Table 11.

The average reduction by using the fuzzy learning method for RMSE, RRSE, RAE and MAE are 25, 13, 15 and 25% respectively as indicated in Table 11.

Table 7 Comparison of parameters (Fuzzy learning model vs. Naïve classifier)

Rule	Naïve Bayes classifier				Fuzzy learning model				Reduction error (%)			
	RMSE	RRSE	RAE	MAE	RMSE	RRSE	RAE	MAE	RMSE	RRSE	RAE	MAE
1	0.41	0.56	0.62	0.28	0.24	0.41	0.41	0.19	41.93	26.79	33.87	32.14
2	0.47	0.6	0.65	0.35	0.32	0.42	0.45	0.24	31.91	30	30.77	31.43
3	0.51	0.64	0.68	0.41	0.37	0.47	0.51	0.29	27.45	26.56	25	29.27
4	0.54	0.68	0.74	0.45	0.41	0.54	0.55	0.34	24.07	20.59	25.68	24.44
5	0.58	0.71	0.75	0.49	0.47	0.59	0.59	0.39	18.97	16.9	21.33	20.41
6	0.62	0.74	0.78	0.57	0.51	0.62	0.64	0.42	17.74	16.22	17.95	26.32
7	0.66	0.77	0.81	0.62	0.55	0.65	0.69	0.46	16.67	15.58	14.81	25.81
8	0.71	0.81	0.84	0.65	0.59	0.71	0.73	0.49	16.9	12.35	13.1	24.62
9	0.75	0.84	0.87	0.74	0.63	0.75	0.77	0.53	16	10.71	11.49	28.38
10	0.79	0.87	0.91	0.78	0.67	0.77	0.81	0.57	15.19	11.49	10.99	26.92
11	0.84	0.91	0.94	0.81	0.71	0.81	0.84	0.61	15.48	10.99	10.64	24.69
12	0.87	0.94	0.96	0.88	0.75	0.84	0.88	0.64	13.79	10.64	8.33	27.27
13	0.91	0.97	0.98	0.95	0.81	0.89	0.91	0.71	10.99	8.25	7.14	25.26
Average	0.67	0.77	0.81	0.61	0.54	0.65	0.68	0.45	20.55	16.7	17.78	26.69

Table 8 Comparison of parameters (Fuzzy learning model vs. Support vector method)

Rule	Support vector method (SVM)				Fuzzy learning model				Reduction error (%)			
	RMSE	RRSE	RAE	MAE	RMSE	RRSE	RAE	MAE	RMSE	RRSE	RAE	MAE
1	0.57	0.52	0.61	0.32	0.24	0.41	0.41	0.19	57.89	21.15	32.79	40.63
2	0.61	0.55	0.64	0.36	0.32	0.42	0.45	0.24	47.54	23.64	29.69	33.33
3	0.65	0.59	0.67	0.39	0.37	0.47	0.51	0.29	43.08	20.34	23.88	25.64
4	0.68	0.62	0.71	0.41	0.41	0.54	0.55	0.34	39.71	12.9	22.54	17.07
5	0.72	0.65	0.74	0.43	0.47	0.59	0.59	0.39	34.72	9.23	20.27	9.3
6	0.74	0.68	0.77	0.46	0.51	0.62	0.64	0.42	31.08	8.82	16.88	8.7
7	0.78	0.71	0.81	0.51	0.55	0.65	0.69	0.46	29.49	8.45	14.81	9.8
8	0.82	0.74	0.84	0.54	0.59	0.71	0.73	0.49	28.05	4.05	13.1	9.26
9	0.85	0.76	0.87	0.57	0.63	0.75	0.77	0.53	25.88	1.32	11.49	7.02
10	0.88	0.79	0.91	0.61	0.67	0.77	0.81	0.57	23.86	2.53	10.99	6.56
11	0.91	0.84	0.93	0.71	0.71	0.81	0.84	0.61	21.98	3.57	9.68	14.08
12	0.94	0.89	0.94	0.81	0.75	0.84	0.88	0.64	20.21	5.62	6.38	20.99
13	0.97	0.93	0.97	0.91	0.81	0.89	0.91	0.71	16.49	4.3	6.19	21.98
Average	0.78	0.71	0.8	0.54	0.54	0.65	0.68	0.45	32.31	9.69	16.82	17.26

Table 9 Comparison of parameters (Fuzzy learning model vs. Regression analysis method)

Rule No.	Regression analysis (RA)				Fuzzy learning model				Reduction error (%)			
	RMSE	RRSE	RAE	MAE	RMSE	RRSE	RAE	MAE	RMSE	RRSE	RAE	MAE
1	0.39	0.42	0.51	0.31	0.24	0.41	0.41	0.19	38.46	2.38	19.61	38.71
2	0.42	0.45	0.54	0.35	0.32	0.42	0.45	0.24	23.81	6.67	16.67	31.43
3	0.45	0.49	0.59	0.39	0.37	0.47	0.51	0.29	17.78	4.08	13.56	25.64
4	0.48	0.54	0.62	0.44	0.41	0.54	0.55	0.34	14.58	0	11.29	22.73
5	0.51	0.61	0.65	0.49	0.47	0.59	0.59	0.39	7.84	3.28	9.23	20.41
6	0.55	0.65	0.69	0.55	0.51	0.62	0.64	0.42	7.27	4.62	7.25	23.64
7	0.58	0.69	0.72	0.57	0.55	0.65	0.69	0.46	5.17	5.8	4.17	19.3
8	0.62	0.73	0.75	0.61	0.59	0.71	0.73	0.49	4.84	2.74	2.67	19.67
9	0.65	0.76	0.78	0.65	0.63	0.75	0.77	0.53	3.08	1.32	1.28	18.46
10	0.7	0.81	0.81	0.71	0.67	0.77	0.81	0.57	4.29	4.94	0	19.72
11	0.79	0.84	0.85	0.76	0.71	0.81	0.84	0.61	10.13	3.57	1.18	19.74
12	0.89	0.88	0.91	0.83	0.75	0.84	0.88	0.64	15.73	4.55	3.3	22.89
13	0.94	0.92	0.94	0.91	0.81	0.89	0.91	0.71	13.83	3.26	3.19	21.98
Average	0.61	0.68	0.72	0.58	0.54	0.65	0.68	0.45	12.83	3.63	7.18	23.41

Table 10 Comparison of parameters (Fuzzy learning model vs. Artificial neural networks)

Rule No.	Artificial neural networks (ANN)				Fuzzy learning model				Reduction error (%)			
	RMSE	RRSE	RAE	MAE	RMSE	RRSE	RAE	MAE	RMSE	RRSE	RAE	MAE
1	0.57	0.59	0.61	0.32	0.24	0.41	0.41	0.19	57.89	30.51	32.79	40.63
2	0.61	0.63	0.64	0.43	0.32	0.42	0.45	0.24	47.54	33.33	29.69	44.19
3	0.65	0.67	0.67	0.47	0.37	0.47	0.51	0.29	43.08	29.85	23.88	38.3
4	0.69	0.71	0.71	0.51	0.41	0.54	0.55	0.34	40.58	23.94	22.54	33.33
5	0.73	0.75	0.74	0.54	0.47	0.59	0.59	0.39	35.62	21.33	20.27	27.78
6	0.78	0.81	0.77	0.57	0.51	0.62	0.64	0.42	34.62	23.46	16.88	26.32
7	0.81	0.84	0.81	0.64	0.55	0.65	0.69	0.46	32.1	22.62	14.81	28.13
8	0.84	0.87	0.84	0.69	0.59	0.71	0.73	0.49	29.76	18.39	13.1	28.99
9	0.87	0.89	0.87	0.77	0.63	0.75	0.77	0.53	27.59	15.73	11.49	31.17
10	0.91	0.91	0.89	0.84	0.67	0.77	0.81	0.57	26.37	15.38	8.99	32.14
11	0.94	0.94	0.91	0.87	0.71	0.81	0.84	0.61	24.47	13.83	7.69	29.89
12	0.96	0.97	0.94	0.91	0.75	0.84	0.88	0.64	21.88	13.4	6.38	29.67
13	0.99	0.99	0.97	0.94	0.81	0.89	0.91	0.71	18.18	10.1	6.19	24.47
Average	0.8	0.81	0.8	0.65	0.54	0.65	0.68	0.45	33.82	20.91	16.52	31.92

Table 11 Summarisation of results of comparison for error reduction

S.No.	Conventional approach	Reduction in error (%) for Fuzzy based approach			
		RMSE	RRSE	RAE	MAE
1	Fuzzy approach versus Naïve	21	17	18	27
2	Fuzzy approach versus SVM	32	10	17	17
3	Fuzzy approach versus Regression analysis	13	4	7	23
4	Fuzzy approach versus Artificial neural networks	34	21	17	32
Average		25	13	14.75	24.75

The average of these values would be 20%. Hence, the fuzzy learning method reduces error compared to conventional approaches by 20%.

5 Conclusion

A fuzzy-based learning approach has been proposed in this work. The fuzzy learning approach reduces the error by 20%. This approach computes the rule strength and chooses rules with rule strengths greater than the threshold limit for effective learning. The learning approach uses a predictive range for the linguistic variable, which is chosen before commencing the training process.

6 Future Work

The work can be extended by making this training process fully unsupervised by generating rule strengths using random function and then completing the entire process. The random unsupervised learning approach could later be implemented using Raspberry Pi as hardware for several applications like healthcare analytics, stock market, etc.

References

1. Taheri, S., Mammadov, M., & Bagirov, A. M. (2011). Improving Naïve Bayes classifier using conditional probabilities. In *9th Australian Data Mining Conference*, pp. 63–69.
2. Liu, H., Yin, X., & Han, J. (2005). An efficient multi-relational Naïve Bayesian classifier based on semantic relationship graph. In *MRDM*, pp. 39–49.

3. Ziddi, N. A., Cerquides, J., Carman, M. J., & Webb, G. I. (2013). Alleviating Naïve bayes attribute independence assumption by attribute weighting. *Journal of Machine Learning Research*, 1947–1988.
4. George, A., Poornachandran, P., & Kaimal, M. R. (2012). adsvm: Pre-processor plug-in using support vector machine algorithm for Snort. ACM 978-1-4503-1822-8/12/08, pp. 179–184.
5. Milchevski, A., Rozza, A., & Taskovski, D. (2015). Multimodal affective analysis combining regularised linear regression and boosted regression trees. ACM 978-1-4503-3743-4/15/10, pp. 33-39, AVEC'15.
6. Zikos, D., & Vandeliwala, I. (2015). Using binary logistic regression coefficients for the dynamic quantification of comorbidities. ACM 978-1-4503-3452-5/15/07 PETRA'15, pp. 12–14.
7. Plagge, M. (2013). Using artificial neural networks to predict first-year traditional students second year retention rates. ACM 978-1-4503-1901-0/13/14, ACMSE'13, pp. 13–18.
8. Trujillo, L., Martinez, Y., & Melin, P. (2011). How many neurons? A genetic programming answer. ACM 978-1-4503-0690-4-11-07/GEECO'11, pp. 175–176.
9. Gunasekara, M., Dharmaratne, A., & Sandaruwan, D. (2014). A feature-point based approach for pose variant face recognition. ACM 978-1-4503-2765-7/14/08, VINCI'14, pp. 242–243.
10. Kalles, D., Verykios, V. S., Feretzakis, G., & Papagelis, A. (2016). Data set operations to hide decision tree rules. ACM 978-1-4503-4304-6/16/08, PrAISe'16, pp. 12–20.
11. Ram, P., & Gray, A. G. (2011). Density estimation trees. ACM 978-1-4503-0813-7/11/08, pp. 627–635, KDD'11.
12. Senior, A., Heigold, G., & Ranzato, M. A. *An empirical study of learning rates in deep neural networks for speech recognition* (pp. 12–17). IEEE.
13. Wang, X. *Method of steepest descent and its applications* (pp. 1–3). IEEE.
14. Li, Y. *Deep reinforcement learning: An overview*. Thesis work: Arxiv, pp. 12–42.
15. Subramanian, D. V., Dr. & Kumar, K. P., Dr. (2016). Fuzzy based modelling for an effective it security policy management. In *SAI Computing Conference 2016*, 978-1-4673-8460-5/16, pp. 1–9. IEEE.