A New Neuro-Fuzzy Classifier based on TSK Model

Jyoti Sharma¹, G. N. Purohit²

¹ Banasthali University, Banasthali – 304022, Rajasthan, India, Mobile No.: +91 9460134346.
² Banasthali University, Banasthali – 304022, Rajasthan, India, Mobile No.: +919828023855.

ABSTRACT

This work presents a Neuro-fuzzy classifier based on Takagi-Sugeno-Kang (TSK) model that train data using scaled conjugate algorithm. The aim of the neuro-fuzzy classified is to optimize the results of the modified k-means algorithm proposed earlier [1]. The previously proposed modified k-means algorithm is two phase algorithm. The first phase identifies actual number of clusters and second phase gives clustering results. Number of clusters and cluster centers are initially provided by the modified K-means algorithm and then apply Takagi-Sugeno-Kang model to generate rule base and finally, speeding up Scaled conjugate gradient algorithm is used to train the model.

Experiments were done on three benchmark datasets IRIS, Thyroid, Wisconsin Diagnostic Breast Cancer and one synthetic dataset to evaluate the performance of proposed neuro-fuzzy classifier.

Key words: Neuro-fuzzy classifier, clustering, scaled conjugate algorithm, TSK model

1. INTRODUCTION

It is very important to obtain more accurate results, due to the rapid advancement of computer technologies and its uses in different areas. The demand of the more accurate results appear in the new scientific research areas such as machine intelligence, data mining, pattern recognition, clustering etc. [2]

Most of the pattern recognition and classification problems usually consist of medium and large datasets. There are many different methods available in literature to solve these problems such as neural networks (NNs) [3], support vector machines (SVM) [4] and Bayes classifier [5] etc. One of the popular networks-based classifier is the neuro-fuzzy classifier (NFC) that combines the learning capabilities of the neural networks as well as powerful description of fuzzy classification techniques [6].

In neuro-fuzzy classifier, fuzzy classification model is used to generate rule-based which is to be trained by neural training methods to get better classification results. There are two main fuzzy classification models available in the literature that are mamdani [7] and Takagi-Sugeno-Kang model [8], [9].There are several training methods proposed in the literature
such as conjugate gradient [10], quasi-newton [11], levenberg-marquardt [12], scaled conjugate gradient [13] etc. some of the modified methods are also there such as speeding up scaled conjugate method [6] etc.

According to Viharo’s [14] survey neuro-fuzzy systems are used in various areas such as monitoring and supervision, fault diagnosis, adaptive filtering, control, pattern identification, system state modeling, variable estimation, variable change detection and damage classification.

The integration of neural networks and fuzzy model have been increased rapidly. Many network training algorithms have been successfully adapted to neuro-fuzzy classifiers to solve the classification problems. The neuro-fuzzy classifier is locally unsupervised and globally supervised. It means the initialization is obtained from the unsupervised clustering methods and then classifier is trained using a supervised method [15].

In the proposed neuro-fuzzy classifier, initialization is obtained from the modified k-means algorithm [1] i.e., initial cluster centers and actual number of clusters. And then, apply TSK model to generate the rule base. A fuzzy rule gives a human understandable expression of the qualitative aspects of the pattern recognition. Finally, classifier is trained using speeding up scaled conjugate algorithm [6]. Details are explained in the section 3.

The reminder of the paper is organized as follows: Section 2 presents state-of-art. Section 3 focuses on the detailed description of proposed methodology. Section 4 depicts experimental analysis and results. Section 5 deals with comparative analysis and Section 6 presents the final conclusion.

2. STATE-OF-THE-ART

2.1. Neuro-fuzzy Classifier (NFC)

The Neuro-fuzzy classifiers are mainly the hybrid learning models. The term neuro-fuzzy classifier is defined as combination of learning of neural networks and human like reasoning style of fuzzy logic. It was proposed by C.T. Sun and J.S. Jang in 1993 [16]. The basic advantage of neuro-fuzzy systems is that they are universal approximators with the ability to solicit interpretable IF-THEN rules [17].

In fuzzy modeling, the neuro-fuzzy classifier is divided into two areas [17]:

(a) The linguistic fuzzy modeling that is focused on interpretability. It is done by the Mamdani model [7].

(b) The precise fuzzy modeling that is focused on accuracy. It is done by the Takagi-Sugeno-Kang (TSK) model [8], [9].

The Neuro-fuzzy classifier is the integration of fuzzy model and neural training algorithm. The fuzzy model gives easily understandable presentation of IF_Then rules and neural network provides fine tuning. Fine tuning is nothing but the training of the classifier to achieve more accurate classification results.

There are several neuro-fuzzy model was proposed in the literature such as ANFIS, FALCON, GARIC, NEFCON, Bayes, K-NN etc. [14],[5],[18]. Some of them are

- Adaptive Network-based Fuzzy Inference System (ANFIS) [19]: ANFIS is one of the most popular neuro-fuzzy models. It is a fuzzy inference system implemented in the framework of adaptive networks. ANFIS can construct input-output mapping between human like reasoning aspects (in the form of fuzzy IF_Then rules with membership functions) and generate stipulated input-output data pairs.

- K-NN [18]: K-Nearest Neighbour first determines the k nearest neighbours using similarity (distance) measure and then identifies the class using those neighbours.
• Bayes classifier [5]: The Bayes classifier uses as determinant functions. It is a probabilistic classifier based on Bayes rules. This classifier assigns most likely class described by its feature vectors.

• NEFClass [21]: It is a NEuro Fuzzy Classification Model. It has a three layer feed forward architecture. The first layer contains the input units (pattern features). The second layer is the hidden layer which contains fuzzy rules and the third layer contains the output for each class. The final output can be calculated by a maximum operation instead of weighted sum.

• Hybrid NFC based on NEFClass Model [21]: FCM was used to initialize the model and trained according to NEFClass training. In this classifier, new rule pruning and structure learning methods were proposed and implemented.

• NFC using Linguistic Hedges [2], [22]: In part 1, the effects of linguistic hedges are shown. And, part 2 presents the fuzzy feature selection model based on linguistic hedges. The values of linguistic hedges are used to show the degree of fuzzy sets.

• An Adaptable Gaussian NFC [23]: It provides automatic classification of data and operates as self-evaluating classifier. In this model, initialization technique partitions, the data set utilizing Gaussian distributions and then merges clusters to produce proper clustering.

• Nonlinear system identification using TSK model [24]: This model generate optimal rule base then identify structure of the model and finally, train the model using Levenberg-Marquardt algorithm.

NFCs are used in various fields such as pattern recognition, food industries, medical diagnosis etc. some of them are as under:

• NFC for cardiac arrhythmias recognition [20]: It is used to identify abnormal beats such as PVC and PAC recording using ECG.

• NFC to identify Sleep stages in Infants [25]: ANFIS based NFC to identify sleep stages in infants.

• Technical diagnosties and Measurement [14]

• Classification of juices [26]: It classifies different juices dataset collected from e-tongue. In this model, first calculate PCA and then apply nearest centroid classifier.

2.2 Takagi-Sugeno-Kang Fuzzy Model (TSK fuzzy model)

The TSK fuzzy model or Sugeno fuzzy model was proposed by Takagi, Sugeno and Kang [8], [9]. This fuzzy model is a systematic approach to generate fuzzy rules using input-output dataset. The sugeno model uses IF_Then fuzzy rules. These fuzzy rules can be written as:

\[
R^k = \text{if } y_1 \text{ is } B_1^k, y_2 \text{ is } B_2^k, \ldots, \text{and } y_m \text{ is } B_m^k \\
\text{then } z^k = a_0^k + a_1^k y_1 + \ldots + a_m^k y_m
\]

Where,

\( m \) is the number of inputs, \( R^k \) is the \( k^{th} \) rule

\( B \) is the fuzzy set with membership function representing fuzzy subspace where \( R \) can be applied for reasoning.

\( y \) is the input linguistic variables

\( y = [y_1, y_2, \ldots, y_m]^T \subset U \in R^m \), \( U \) is the input universe of discourse.

\( z \) is the output variable, \( z \subset Z \in R \), \( Z \) is the output universe of discourse.

The final output is the weighted average output.

\[
Z = f(y) = \frac{\sum_{k=1}^{n} z^k \delta^k}{\sum_{k=1}^{n} \delta^k}
\]
Where,
\[ z^k = \alpha_0^k + \sum_{l=1}^n \alpha_l^k y_l \tag{3} \]
and
\[ \delta = \prod_{j=1}^m B_j^k (y_j^0) \tag{4} \]

This model is a multi-input and single-output fuzzy model. It is capable of describing nonlinear system with less number of rules [24]. TSK fuzzy model is used to identify parameters and structure.

3. PROPOSED METHODOLOGY

3.1. Scaled Conjugate Algorithm

The scaled conjugate algorithm [13] was introduced by Moller in 1993. It is fully automated and faster that the standard back propagation algorithm [2], the conjugate gradient algorithm with line search (CGL) and the one-step Broyden-Fletcher-Goldfarb-Shanno memory less quasi-Newton algorithm (BFGS) [5]. Bayram Cetisli and Atalay Barkana introduced speeding up of scaled conjugate gradient algorithm (SSCG) [6] in 2010 which speeding up the execution time of the algorithm. This work used speeding up scaled conjugate gradient algorithm to reduce the execution time.

The SSCG algorithm is given as:

1. Choose the initial values of the parameters \( \partial_k \) and \( 0 < a < 1 \). Set \( k = 1 \). Then, calculate the gradient and the conjugate direction of \( \partial_k \):
   \[ b_{k-2} = -y_{k-2} = -E'(\partial_{k-2}) \tag{5} \]
   and set success = true.

2. The calculation of the short step of the \( (k - 1) \)th iteration operations is as follows:
   \[ a_{k-1} = \frac{a}{|b_{k-2}|} \tag{6} \]
   \[ \partial_{k-1} = \partial_{k-2} + a_{k-1} b_{k-2} \tag{7} \]
   and \( y_{k-1} = E'(\partial_{k-1}) \tag{8} \)

3. Calculate the long step size of the \( k \)th iteration:
   then find its gradient and the new real point \( \theta_k \):
   \[ y_k = E'(\partial_k) \tag{9} \]
   \[ s_k = \frac{E'(\partial_{k-2}) - E'(\partial_{k-1})}{a_{k-1}} \tag{10} \]
   \[ \alpha_k = \frac{b_k^T E'(\partial_k)}{b_k^T s_k} \tag{11} \]
   \[ \partial_k = \partial_{k-2} + \alpha_k b_{k-2} \tag{12} \]
   Set for the next gradient estimation, \( g, X \) and \( Y \) using the above obtained results

\[ \text{For } i = 1 \text{ to } M \]
\[ X_i = \begin{bmatrix} \partial_{k-2}(i) \\ \partial_{k-1}(i) \\ \partial_k(i) \end{bmatrix} = \begin{bmatrix} \partial_{k-2}^2(i) & \partial_{k-2}(i) & 1 \\ \partial_{k-1}^2(i) & \partial_{k-1}(i) & 1 \\ \partial_k^2(i) & \partial_k(i) & 1 \end{bmatrix}, \text{ and} \]
\[ Y_i = [y_{k-2}(i) \quad y_{k-1}(i) \quad y_k(i)]^T. \tag{13} \]

4. Set success = true, and
   \[ b_k = -y_k = -E'(\partial_k) \tag{15} \]

5. Determine the temporary point and its gradient as follows:
\[ a_k = a, \partial_{t,k} = \partial_k + a_k b_k. \] (16)

For \( i = 1 \) to \( M \)

\[ X_i F_i = Y_i. \] (17)

If \( X_i \) is non–singular and has an inverse, then

\[ F_i = X_i^{-1} Y_i. \] (18)

Otherwise, if \( X_i \) is singular or has no inverse, then

\[ F_i = (X_i^T X_i)^{-1} X_i^T Y_i. \] (19)

\[ E^T \left( \partial_{t,k}(i) \right) = \left[ \begin{array}{cc} \left( (\partial_{t,k}(i))^2 \right) & \partial(i) \\ \end{array} \right]^T F_i. \] (20)

\[ y_{t,k}^e(i) = -E^T (\partial_{t,k}(i)) \] (21)

\[ 6. \] The second-order information is calculated only if success = true

\[ s_{f,k} = \frac{y_{t,k}^e(i) - y_k}{a_k}, \] (22)

\[ \delta_k = b_k^T s_{f,k}. \] (23)

7. Calculate scale factor \( s_{f,k} \):

\[ s_{f,k} = s_{f,k} + (l_k - \bar{l}_k) b_k, \] (24)

\[ \delta_k = \delta_k + (l_k - \bar{l}_k) |b_k|^2. \] (25)

8. If \( \delta_k \leq 0 \), then the Hessian matrix must be made positive-definite:

\[ s_{f,k} = s_{f,k} + (l_k - 2 \frac{\delta_k}{|b_k|^2}) b_k, \] (26)

\[ \bar{l}_k = 2(l_k - \frac{\delta_k}{|b_k|^2}), \] (27)

\[ \delta_k = -\delta_k + l_k |b_k|^2, \] (28)

\[ l_k = l_k. \] (29)

9. Calculate step size \( \alpha_k \) and the new real point \( \partial_{k+1} \)

\[ \tau_k = b_k^T y, \] (30)

\[ \alpha_k = \frac{\tau_k}{\delta_k}. \] (31)

\[ \partial_k = \partial_k + \alpha_k b_k. \] (32)

10. The calculation of the reference for comparison:

\[ u_k = \frac{2 \delta_k (E(\partial_k) - E(\partial_{k+1}))}{\tau_k}. \] (33)

11. If \( u_k \geq 0 \), then minimization of the cost function is achieved:

\[ y_{k+1} = E'(\partial_{k+1}), \] (34)

\[ \lambda_k = 0 \] (35)

and success = true.

- Add the new values \( \partial_{k+1} \) and \( y_{k+1} \) into \( X \) and \( Y \) below, and then, the first rows of \( X \) and \( Y \) are removed.

- If \( (k \mod M) = 0 \), then the algorithm is restarted:

\[ b_{k+1} = -y_{k+1}. \] (36)

else, a new conjugate direction is created:

\[ \beta_k = \frac{|y|^2 - y_{k+1} y_k}{\tau_k}, \] (37)

\[ b_{k+1} = -y_{k+1} + \beta_k b_k. \] (38)

- If \( u_k \geq 0.75 \), then the scale factor is decreased:

\[ l_k = 0.5 l_k. \] (39)

else, minimization of the cost function is impossible:

\[ \bar{l}_k = l_k. \] (40)

and success = false.
12. If \( u_k < 0.25 \), then the scale factor is increased:
\[
l_k = 4l_k. \tag{41}
\]

13. If the steepest direction \( y_k \neq 0 \), then set \( k = k + 1 \) and go to step 5;
else, the algorithm is completed and
\[
\partial_{k+1} \text{is the desired minimum point of the cost function.}
\]

The first three steps are used for initialization and to estimate the new gradient. These three steps are executed once and actual calculation or repetition of steps to train dataset is started from the fourth step. For more details, refer reference no. [6].

3.2. Modified K-Means algorithm

Modified k-means [1] is a two phased algorithm. First phase identifies correct number of clusters and second phase is the fine clustering. The number of clusters identifies using validity index \( VI (m, \lambda) \). This validity index need not require range of number of clusters. It starts calculating from two clusters and increased by one every time up to the actual number of clusters. Because of this friendly behavior of the validity index, the fine clustering start automatically after getting actual number of clusters. The second phase is the modified k-means algorithm to increase the classification accuracy. The fine clustering phase is repeated by \( \alpha \) times.

The steps of the pseudo algorithm are given below:

**Step 1**: Initialize \( noc = 2 \)

**Step 2**: Normalize Data Set \( Y \)
\[
Y_{ij} = \frac{Y_{ij} - Y_{\min}}{Y_{\max} - Y_{\min}} \tag{42}
\]

**Step 3**: Identify Number of Clusters (Phase I)

**Step 3.1**: Loop

**Step 3.2**: Run K-means for \( noc \) clusters

**Step 3.3**: Calculate proposed validity index, \( VI (m, \lambda) \)
\[
VI (m, \lambda) = 1 - \left( \frac{k}{j} \right) * (\sigma^2 (m)) \tag{43}
\]

\( VI (m, \lambda) \) is calculated for each cluster.

\( \sigma^2 (m) \) is the variance of membership value = \[
\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} \left( m_{ij} - \bar{m}_{ij} \right)^2 \tag{44}
\]

and \( \lambda \) is defined as

If \( N > f \),
\[
\lambda = \frac{f \cdot noc^2}{N} \tag{45}
\]

Otherwise,
\[
\lambda = \frac{2 \cdot N \cdot noc^2}{f} \tag{46}
\]

Where, \( noc \) is the assumed number of clusters, \( N \) is the number of data vectors and \( f \) is the number of features.

**Step 3.4**: If \( VI (m, \lambda) \leq 0 \) then \( noc = noc + 1 \) else terminate the loop.

The optimal number of clusters is \((noc - 1)\).

**Step 4**: Fine Clustering (Phase II)

**Step 4.1**: for \( i = 1 \) to \( \alpha \)

**Step 4.2**: Calculate the minimum difference \( (\text{min}_\text{diff}) \) from the distance between two nearest clusters of each data point.

**Step 4.3**: Update two nearest cluster centers using below given equation
\[
C_{ij} = C_{ij} \pm \theta \tag{47}
\]

**Step 4.4**: Calculate Euclidean distance
\[
D_{ij} = \sum_{i=1}^{n} \sum_{j=1}^{noc} \| y_i - C_j \|^2 \tag{48}
\]

from the new centers & update partition matrix.

**Step 4.5**: Calculate cluster centers
\[ C_{ij} = \sum_{i=1}^{n_{oc}} \sum_{j=1}^{m} \frac{m_{ij}^* x_j}{m_{ij}} \] 

**Step 4.6:** If cluster center becomes constant then break for loop otherwise continue loop

**Step 5:** Display clustering result

### 3.3. Proposed Neuro-Fuzzy Classifier

The proposed neuro-fuzzy classifier combines modeling and training. It has three phases: Initialization, TSK model and SSCG training algorithm. The initialization is first obtained from the modified k-means algorithm [1] is returns actual number of clusters and cluster centers. Then, TSK model is used to generate the rule base. A fuzzy rule gives a human understandable expression of the qualitative aspects of the pattern recognition. And finally apply speeding up scaled conjugate algorithm to perform fine-tuning.

The steps of the proposed model are as follows:

**Step 1:** *Initialization:* Obtain Initial fuzzy model from modified k-means.

**Step 2:** *Modeling:* Apply TSK model to generate rule base.

**Step 3:** *Fine tuning:* The classifier is trained using speeding up scaled conjugate algorithm.

The procedural flow of the proposed neuro-fuzzy classifier is as below:

(i) The first step is to load the dataset such as iris, thyroid dataset which is to be trained using proposed neuro-fuzzy classifier.

(ii) The second step is to initialize the number of clusters and cluster centers which are identified by executing modified k-means [1].

(iii) The third step applies fuzzy classifier based on sugeno classifier. The Sugeno model is popular, simple and transparent model. This model returns rule-base. This rule-base is composed of simple if-then rules.

(iv) The fourth step is to train the fuzzy classifier using speeding up scaled conjugate gradient training algorithm. The classifier returns class label. These class labels tell that the data vectors belong to which class.

(v) And, the final step evaluates the performance of the classifier with the help of confusion matrix.

The flow chart is shown in the figure 1.
EXPERIMENTAL ANALYSIS AND RESULTS

In this section, the performance of the neuro-fuzzy classifier was evaluated by applying it to the three well-known real datasets IRIS, Thyroid and Wisconsin Diagnostic Breast Cancer using confusion matrix and one synthetic dataset.

4.1. Datasets

Data 1: IRIS data set is the widely used four featured dataset. These four features are sepal length, sepal width, petal length and petal width. The dataset contains 150 data vectors of three classes named as Virginica, Setosa and Veriscolor. Setosa and Veriscolor are overlapped classes. Each class contains 50 data points.
Table 1
Comparison of clustering result of proposed work for iris dataset

<table>
<thead>
<tr>
<th>Actual Class Label</th>
<th>Proposed work</th>
<th>Modified K-means [1]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Predicted Class Label</td>
<td>Predicted Class Label</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>49</td>
</tr>
</tbody>
</table>

Figure 2: Root Mean Square Error of 300 epochs for IRIS dataset

The proposed neuro-fuzzy classifier trained with 99.33% accuracy with average RMSE after 300 epochs is 0.0200351. The classifier misclassifies only one data vector and rest of the data vectors classify correctly. It increases 8% accuracy of the modified k-means. The root mean square errors of different epochs are shown in figure 2 and confusion matrix in Table 1.

Data 2: Thyroid database contains 215 data points. Each data point has five features. The results of the tests are considered as classes. There are three classes which are normal, hyper and hypo. Normal class has 150 instances; hyper has 35 and 30 instances which are of hypo class.

The proposed neuro-fuzzy classifier trained with 99.53% accuracy. The classifier classifies 214 data vector correctly out of 215 data vectors. That means it misclassifies only one data. The classification accuracy increases from 90.70% to 99.53% i.e., 8.83%. The average RMSE after 300 epochs is 0.0139501. The confusion matrix is shown in Table 2 and the root mean square errors of different epochs are shown in figure 3.
Table 2
Comparison of clustering result of proposed work for thyroid dataset

<table>
<thead>
<tr>
<th>Actual Class Label</th>
<th>Proposed work</th>
<th>Modified K-means [1]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Predicted Class Label</td>
<td>Predicted Class Label</td>
</tr>
<tr>
<td>1</td>
<td>149</td>
<td>150</td>
</tr>
<tr>
<td>2</td>
<td>35</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>23</td>
</tr>
</tbody>
</table>

Figure 3: Root Mean Square Error of 300 epochs for Thyroid dataset

Data 3: Wisconsin Diagnostic Breast Cancer (WDBC) dataset contains 569 data points with 30 features. This dataset contains data of two different classes. 212 data points are from one class and 357 data points from the another class.

Table 3
Comparison of clustering result of proposed work for WDBC dataset

<table>
<thead>
<tr>
<th>Actual Class Label</th>
<th>Proposed work</th>
<th>Modified K-means [1]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Predicted Class Label</td>
<td>Predicted Class Label</td>
</tr>
<tr>
<td>1</td>
<td>206</td>
<td>180</td>
</tr>
<tr>
<td>2</td>
<td>349</td>
<td>351</td>
</tr>
</tbody>
</table>
The proposed neuro-fuzzy classifier trained with 97.54% accuracy after 1200 epochs and 97.18% after 300 epochs. The classifier misclassifies 14 data vectors out of 569 data vectors and rest of the data vectors classify correctly. It increases 4.22% accuracy of the modified k-means. The confusion matrix is shown in Table 3 and the root mean square errors of 1200 epochs are shown in Figure 4. The average RMSE after 1200 epochs is 0.0637218.

*Data 4:* D2C5 is the synthetic dataset. There are 250 data vectors in this two-dimensional dataset. This dataset is divided into five equal spherical clusters i.e., 50 data vectors in each class. Details of the dataset are available in reference no [1]. The proposed work attains 100% classification accuracy. The confusion matrix is shown in Table 4.

**Table 4**
Confusion matrix of D2C5 dataset

<table>
<thead>
<tr>
<th>Actual Class → Predicted Class ↓</th>
<th>Class I</th>
<th>Class II</th>
<th>Class III</th>
<th>Class IV</th>
<th>Class V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class I</td>
<td>50</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Class II</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Class III</td>
<td>0</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Class IV</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>Class V</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>50</td>
</tr>
</tbody>
</table>
4.2 Comparative analysis

Accuracy rate of pseudo-algorithm is compared with some of the available classifiers. Comparison of classification accuracy for IRIS dataset, thyroid dataset, WDBC dataset and D2C5 dataset are shown in Table 5, Table 6, Table 7 and Table 8 respectively. The highest accuracy rate is mentioned in bold. According to the below given comparison if classification accuracy is summarized as

(i) The proposed algorithm attains highest accuracy for IRIS, thyroid and D2C5 datasets and second highest accuracy for WDBC dataset.

(ii) RBF (training) [27] attain highest accuracy for WDBC dataset i.e., 98.93% and the proposed work attains 97.54% accuracy.

(iii) The proposed work attains 100% accuracy for the D2C5 dataset.

(iv) The IRIS dataset and thyroid dataset accuracy hikes up to 99.33% and 99.53% respectively.

Table 5
Comparisons of classification accuracy rate of proposed neuro-fuzzy classifier for IRIS dataset

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed work</td>
<td><strong>99.33%</strong></td>
</tr>
<tr>
<td>Modified K-means [1]</td>
<td>91.33%</td>
</tr>
<tr>
<td>Trainable Fuzzy System [28]</td>
<td>96.00%</td>
</tr>
<tr>
<td>Histogram based fuzzy system [28]</td>
<td>97.33%</td>
</tr>
<tr>
<td>ANCIS/mountain clustering [28]</td>
<td>97.33%</td>
</tr>
<tr>
<td>NEFClass [28]</td>
<td>96.67%</td>
</tr>
<tr>
<td>Fuzzy Kohonen Net [28]</td>
<td>91.33%</td>
</tr>
<tr>
<td>KNN [28]</td>
<td>96.67%</td>
</tr>
<tr>
<td>Bayes [28]</td>
<td>97.33%</td>
</tr>
</tbody>
</table>

Table 6
Comparisons of classification accuracy rate of proposed neuro-fuzzy classifier for Thyroid dataset

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed work</td>
<td><strong>99.53%</strong></td>
</tr>
<tr>
<td>Modified K-means [1]</td>
<td>90.70%</td>
</tr>
<tr>
<td>ANFIS [29]</td>
<td>71.40%</td>
</tr>
<tr>
<td>Fuzzy-MLP [29]</td>
<td>88.53%</td>
</tr>
<tr>
<td>MLP [29]</td>
<td>90.09%</td>
</tr>
<tr>
<td>Fuzzy-RBF [29]</td>
<td>81.54%</td>
</tr>
<tr>
<td>RBF [29]</td>
<td>65.32%</td>
</tr>
<tr>
<td>Fuzzy-CSFNN [29]</td>
<td>92.93%</td>
</tr>
<tr>
<td>CSFNN [29]</td>
<td>83.91%</td>
</tr>
</tbody>
</table>
Table 7
Comparisons of classification accuracy rate of proposed neuro-fuzzy classifier for WDBC dataset

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed work (300 epochs)</td>
<td>97.18%</td>
</tr>
<tr>
<td>Proposed work (1200 epochs)</td>
<td>97.54%</td>
</tr>
<tr>
<td>Modified K-means [1]</td>
<td>93.32%</td>
</tr>
<tr>
<td>RBF (training) [27]</td>
<td>98.93%</td>
</tr>
<tr>
<td>RBF (testing) [27]</td>
<td>97.37%</td>
</tr>
<tr>
<td>FRNN (training) [27]</td>
<td>92.38%</td>
</tr>
<tr>
<td>FRNN (testing) [27]</td>
<td>92.94%</td>
</tr>
<tr>
<td>FRNN_FS (training) [27]</td>
<td>93.70%</td>
</tr>
<tr>
<td>FRNN_FS (testing) [27]</td>
<td>95.88%</td>
</tr>
</tbody>
</table>

Table 8
Comparisons of classification accuracy rate of proposed neuro-fuzzy classifier for D2C5 dataset

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed work (300 epochs)</td>
<td>100%</td>
</tr>
<tr>
<td>Modified K-means [1]</td>
<td>97.20%</td>
</tr>
</tbody>
</table>

5. CONCLUSIONS

The proposed model is a neuro-fuzzy approach because it models the system using one of the popular fuzzy models i.e. Takagi-Sugeno-Kang model and train the model using modified version of popular network training algorithm scaled conjugate algorithm. This approach gives us a way to train datasets without pre-specifying number of clusters. Modified k-means [1] returns number of clusters and cluster centers that are used as initial parameters of the model and then apply fuzzy classifier to generate rule base and finally apply speeding up scaled conjugate algorithm [6] algorithm to train dataset properly. It returns class labels that specify the data vector belongs to which class. So, the confusion matrix is used to identify the classification accuracy.

Experiments performed on three real datasets and an artificial dataset proved that proposed neuro-fuzzy classifier trained clustering results well.

REFERENCES