An Unsupervised Feed Forward Neural Network Method for Efficient Clustering

Roya Asadi¹, Mitra Asadi¹, and Shokoofeh Asadi²
¹Central Tehran Branch, Islamic Azad University, Iran
²Department of Agricultural Management Engineering, University of Science and Research, Iran

Abstract: This paper presents a Real Unsupervised Feed Forward Neural Network (RUFFNN) clustering method with one epoch training and data dimensionality reduction ability to overcome some critical problems such as low training speed, low accuracy as well as high memory complexity in this area. The RUFFNN method trains a code book of real weights by utilizing input data directly without using any random values. The Best Match Weight (BMW) vector is mined from the weight codebook and consequently the Total Threshold (TT) of each input data is computed based on the BMW. Finally, the input data are clustered based on their exclusive TT. For evaluation purposes, the clustering performance of the RUFFNN was compared to several related clustering methods using various data sets. The accuracy of the RUFFNN was measured through the number of clusters and the quantity of Correctly Classified Nodes(CCN). The superior clustering accuracies of 96.63%, 96.67% and 59.36% were for the breast cancer, iris and spam datasets from the UCI repository respectively. The memory complexity of the proposed method was \(O(m.n.s_a)\) based on the number of nodes, attributes and size of the attribute.

Keywords: Artificial neural network, feed forward neural network, unsupervised learning, clustering, real weight.

Received December 2, 2014; accepted August 16, 2015

1. Introduction

A Feed Forward Neural Network (FFNN) is a popular tool for statistical decision making and inspired by the brain task. In this network, processing of data has only one forward direction from the input layer to the output layer without any cycles or backward [1, 12]. Learning is an important property of the neural network.

2. Related Works

According to the study of current Unsupervised Feed Forward Neural Network (UFFNN) clustering methods [4], Vector Quantization (VQ) [16], K-means [11] and some UFFNN clustering methods such as Kohonen’s Self-Organizing Map (SOM) [16], Neural Gas (NG) [17] and Growing Neural Gas (GNG) [9] are considered as fundamental patterns in the current UFFNN clustering methods in the static and online dynamic environments.

The VQ gains a suitable codebook of the weights for clustering based on probability density functions. K-means is a partitioning clustering method by using Centroid-Based technique similar to the VQ, however, it should define a number of clusters and parameters before clustering [2, 13, 18]. NG is based on the VQ and data compression. The NG dynamically partitions itself like gas and describes the number of clusters, but it cannot control the growth of the network of nodes. The GNG method, on the other hand, is able to follow dynamic distributions by adding nodes and deleting them in the network during clustering. First, two random nodes from the input data are selected and the network competition is started for the highest similarity to the input pattern. During the learning related data nodes are classified as similarities within clusters, however, the number of nodes is increased in order to get the input probability density [2, 13, 18]. SOM maps multi-dimensional data onto lower dimensional subspaces where the geometric relationships between points indicates their similarity, based on a competitive learning and adjusting the weights to be close to the “winning” nodes, by using enough and necessary data in order to develop meaningful clusters [10, 16]. Current UFFNN clustering methods have major problems in low speed and low accuracy of clustering with the high memory complexity of the network, some reasons are [2, 13, 18]: initialization the weights, thresholds and parameters for controlling clustering tasks by using random values, and the suitable values are defined through trial and error after several re-performances of the model and relearning; high dimensional data and huge datasets which cause difficulty in managing new data and noise while pruning causes data details to be lost [20].
3. Methodology

In order to solve the aforementioned problems, we proposed a Real Unsupervised Feed Forward Neural Network (RUFFNN) clustering. Figure 1 shows its design.

1. **Data Pre-Processing**: mostly pre-processing is the contributing factor in developing efficient techniques for low training time and high accuracy of feed forward neural network clustering [4].

In this study, the MinMax normalization as a data pre-processing technique [4, 13] is used in order to consider the input value in the range (0, 1). Therefore, each attribute $X_{ij}$ is normalized according to Equation 1:

$$X_{ij} = \frac{X_{ij} - \text{Min}(X_{ij})}{\text{Max}(X_{ij}) - \text{Min}(X_{ij})} \times (1-0)+0$$  

(1)

2. **Creating a Real Weights Code Book**: the proposed method is based on the SOM and torque vector [3, 21], however, it generates a code book of non-random weights. Each attribute of the input data $X_{ij}$ is normalized based on the Standard Gaussian Distribution (SGD), as shown in Equation 2:

$$SGD(X_{ij}) = \frac{X_{ij} - \mu_j}{\sigma_j}$$  

(2)

$\mu_i$ and $\sigma_i$ are mean and standard deviation of the input data record. Each SGD$(X_{ij})$ shows the distance of each input value of each instance (record) from the mean of the input data $X_i$. The RUFFNN method considers each SGD$(X_{ij})$ as the weight $W_{ij}$ for that value as shown in Equation 3, therefore, each weight vector of the codebook is computed based on processing on each input data instance and is not at random. This phase can be processed in parallel.

$$W_{ij} = SGD(X_{ij}); \ i = 1, 2, ..., n; \ j = 1, 2, ..., m$$  

(3)

3. **Mining the Best Match Weight (BMW) Vector from the Codebook**: the BMW vector is the global geometric mean [14] vector of the weights code book. The BMW consists of the components $WGM$ for attributes as geometric mean weight. The $WGM_j$ is computed by taking the $n$th root of the product of the real weights of each attribute of the input data. The parameter $n$ is the number of input data, $i$ is current number of node of input data; $m$ is the number of attributes and $j$ is the current number of the attribute of input data, as shown in Equations 4 and 5.

$$WGM_j = (W_{ij1} \cdot W_{ij2} \cdots W_{ijm})^{1/n}$$  

(4)

$$BMW = (WGM_1, WGM_2, ..., WGM_m)$$  

(5)

The RUFFNN method tries to learn the BMW vector as the criterion weight vector. For example, the BMW vectors of the Breast Cancer Wisconsin (BCW) and Iris datasets from UCI [5] are as follows:

- **BMW of BCW**:
  - $(0.132488, 0.095497, 0.096868, 0.10093, 0.097176, 0.141244, 0.105377, 0.107894, 0.122526);
- **BMW of Iris**:
  - $(0.159178919, 0.3448055, 0.205496179, 0.290519403)$

4. **Fine Tuning**: this process refers to modifying the weights accurately in order to succeed better results of clustering the input data, as follows:

- **Smoothing the Weights**: one of the smoothing technique includes flexible and robust parameters of the FFNN clustering tasks is the weights interconnection to improve speed, accuracy and capability of the training and optimization [11, 22]. **Mid-range** is a popular smoothing technique [11, 22]. Some attributes of the input data have too high weight amounts which may cause them to overlook the high thresholds and high effect the results of clustering. Therefore, when some components of the BMW vector are extremely higher than the other components, the **Mid-range** technique can be used. In the **Mid-range** technique, the average of the high weight components of the BMW vector is calculated and considered as the Middle range (Mid-range). If some components of the BMW vector are upper
than the *Mid-range*, the method fixes their weights to the *Mid-range* value.

- **Data Dimension Reduction**: high dimensional data and huge data set cause difficulty in handling new data and noise while pruning causes data details to be lost [2, 15]. The RUFFNN method reduces the dimension of data by recognizing the weak and ineffective WGM$_j$ and removing the related attributes, that it affects high speed and low network memory usage complexity [2, 7]. Hence, the weights can be controlled and pruned in advance.

5. **Single Layer FFNN Clustering**: The topology is very simple as Figure 1 shows, it contains of just an input layer with $n$ nodes equal the number of attributes and an output layer with just one node. The units of the input layer are fed by the normalized data values from the data pre-processing phase. Each unit applies a related weight component WGM$_j$ of the *BMW* vector. The output layer consists of one unit with a weighted sum function for computing a threshold as the actual desired output. The RUFFNN training is carried out just in one epoch and is based on real weights, without any class label, parameter for controlling tasks, weight updating, activation function and error function such as mean square error. Due to apply the geometric mean of the weights for computing the *BMW*, the range and properties of the input data values cannot dominate the values of the thresholds. Figure 2 shows an example about each $X_i$ which creates its own torque vector [3, 21] ratio to the global mean or the gravity center of the dataset.

Each $X_i$ by using WGM$_j$ as its arm, shows the distance of $X_i$ from the gravity center of the matrix, and creates a torque vector, which is equal the threshold $T_{ij}$ as shown in Equation 6.

$$T_{ij} = X_{ij} \cdot \text{WGM}_j$$

The vectors are evaluated together and eventually will reach the equilibrium. After equivalence, the Total Threshold (TT) of each $X_i$ is computed, as shown in Equations 7 and 8.

$$TT_i = \sum_j X_{ij} \cdot \text{WGM}_j; \quad j=1,2,...,m$$

$$OR\quad TT_i = \sum_j T_{ij}; \quad j=1,2,...,m$$

- **Clustering of the Input Data**: each vector value of the $X_i$ takes place its own position on the torque axis. Therefore, the input data based on their exclusive TT lay on the torque axis respectively, and has an exclusive and individual threshold. If there are two input data with the equally TT, but different clusters or classes, as error of the clustering method, decrease the clustering accuracy. The RUFFNN considers the input data with near TT into one cluster. After clustering, each cluster will be delegated to the special class which is most frequent in the cluster. Figure 3 shows the BCW dataset from UCI repository [5] which is clustered to two clusters by the RUFFNN. The input data point A has $TT_A=0.004299222$ and lies inside of the cluster1 or the cluster of the “Malignant”.

- **Pruning the Noise**: The RUFFNN distinguishes isolated input data point through solitary TT, which is not near the TTs of other clustered data point. Therefore, the isolated data lies out of the locations of other clusters. The proposed method sets apart these data points as noise and removes them. The action of removing the noise causes high speed and clustering accuracy with low memory usage of the network.

During training, all phases of the proposed method must be performed completely, however, during testing just by having normalized data values of the test set and the *BMW* components from training phase, the single layer FFNN is able to cluster the of the test data, immediately. The algorithm of the RUFFNN clustering method is, as follows:

*Input*: Data set $N$; **Output**: Clusters of dataset;  
Initialize the parameters:

1. Input a new vector $X_i$; 
2. 
3. //Data preprocessing : For all $X_i = \text{Normalized}(X_i)$;
//Compute the weight vector for all X_i; 
    SGD(X_i) = (X_i - μ)/σ_i;  W_i = SGD(X_i);
}
2- // Mining the Best Match Weight vector (BMW);
    Each WGM_j=(W_j1, W_j2, ... W_jm); 
    BMW= (WGM_1, WGM_2, ..., WGM_n);
3- // Fine tuning through two techniques: Smoothing the components of the BMW, and Data dimension reduction;
    For all components of the BMW: Mid-range(WGM_j) ; 
    Delete attributes with weak WGM_j;
4- // Process of the single layer UFFNN clustering 
    // Compute the Total Threshold of each input data X_i; 
    TT_i = ∑X_i j WGM_j; 
    Delete isolated input data as noise with solitary TT; 
    Clustering; Group the X_i with similar TT in one cluster; }

4. Experimental Results and Comparison

The methods were implemented in Visual C#.Net under Microsoft Windows 7 Professional operating system with 4 GHz Pentium processor. The proposed method was tested for the BCW, Iris and Spambase datasets from the UCI repository [5], as shown in Table 1. The accuracies of the methods were measured through the number of clusters and the quantity of the Correctly Classified Nodes (CCN), which was equal true positive and true negative nodes [6, 8], and showed the total of nodes and their densities, with the correct class in the correct related cluster, in all created clusters by the method. Furthermore, the accuracy was also measured by using the F-measure function with 10 folds of the test set. The results were the averages of three time performances.

Table 1. The information of selected datasets in this study from the UCI Repository.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Data Set Characteristics</th>
<th>Attribute Characteristics</th>
<th>Number of Instances</th>
<th>Number of Attributes</th>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast Cancer Wisconsin (Original)</td>
<td>Multi variable</td>
<td>Integer</td>
<td>699</td>
<td>10</td>
<td>Two classes: benign and malignant</td>
</tr>
<tr>
<td>Iris</td>
<td>Multi variable</td>
<td>Real</td>
<td>150</td>
<td>4</td>
<td>Three classes: Iris Setosa, Iris Versicolour and Iris Virginica</td>
</tr>
<tr>
<td>Spambase</td>
<td>Multi variable</td>
<td>Integer-Real</td>
<td>4601</td>
<td>57</td>
<td>Two classes: Spam and Non-Spam</td>
</tr>
</tbody>
</table>

4.1. Breast Cancer Wisconsin Data Set

The BCW dataset is clustered by the RUFFNN, as shown in Figure 3, and the result is compared with the related methods [6, 8], as shown in Table 2. That, the SOM produced 660 CCN after 20 epochs with 96.63% density of CCN, as the best UFFNN result. The RUFFNN after one epoch in 8.7262 milliseconds, had 660 CCN, 96.63% density of CCN and 98.06% of the F-measure. All clustering methods show two clusters for this dataset. The Back Propagation Network (BPN) [23], as a popular supervised FFNN, classified this dataset after 1000 epochs with 99.28% by F-measure accuracy.

Table 2. Comparison of clustering the BCW data set by different UFFNN methods.

<table>
<thead>
<tr>
<th>The Clustering Method</th>
<th>CCN</th>
<th>Density of CCN %</th>
<th>Epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOM</td>
<td>660</td>
<td>96.63</td>
<td>20</td>
</tr>
<tr>
<td>K-Means</td>
<td>657</td>
<td>96.19</td>
<td>20</td>
</tr>
<tr>
<td>Neural Gas</td>
<td>657</td>
<td>96.19</td>
<td>20</td>
</tr>
<tr>
<td>GNG</td>
<td>477</td>
<td>69.84</td>
<td>5</td>
</tr>
<tr>
<td>RUFFNN</td>
<td>660</td>
<td>96.63</td>
<td>1</td>
</tr>
</tbody>
</table>

4.2. Iris Data Set

The Iris dataset is clustered by the RUFFNN, as shown in Figure 4, and the result is compared with the related methods [6, 8], as shown in Table 3. That, the CCN of the NG was 139 after 20 epochs with 92.67% density of CCN, as the best UFFNN result [7]. The RUFFNN after one epoch in 4.1744 milliseconds, had 145 CCN, 96.67% density of CCN, and 97.33% of the F-measure.

![Iris dataset clusters](image)

Table 3. Comparison of clustering the Iris data set by different UFFNN methods.

<table>
<thead>
<tr>
<th>The Clustering Method</th>
<th>CCN</th>
<th>Density of CCN %</th>
<th>Epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOM</td>
<td>123</td>
<td>82.00</td>
<td>20</td>
</tr>
<tr>
<td>K-Means</td>
<td>134</td>
<td>89.33</td>
<td>20</td>
</tr>
<tr>
<td>Neural Gas</td>
<td>139</td>
<td>92.67</td>
<td>20</td>
</tr>
<tr>
<td>GNG</td>
<td>135</td>
<td>90.00</td>
<td>10</td>
</tr>
<tr>
<td>RUFFNN</td>
<td>145</td>
<td>96.67</td>
<td>1</td>
</tr>
</tbody>
</table>

All clustering methods show three clusters for this dataset. The BPN classified this dataset after 14 epochs with 94% by the F-measure accuracy.

4.3. Spambase Data Set

The Spambase dataset is clustered by the RUFFNN, as shown in Figure 5, and the result is compared with the related methods [6, 8], as shown in Table 4. That, the SOM produced 1210 CCN after 20 epochs with 26.30% density of CCN, as the best UFFNN results. The RUFFNN clusters this dataset, after one epoch in 337.1057 milliseconds, with 2731 CCN, 59.36% density of CCN, and 66.46% of the F-measure. All clustering
methods show two clusters for this dataset. The BPN classified the Spambase after 2000 epochs with 80% accuracy by the F-measure.

![Image](spambase.png)

Figure 5. The clusters of the Spambase dataset by the RUFFNN method.

Table 4. Comparison of clustering the Spambase dataset by different UFFNN methods.

<table>
<thead>
<tr>
<th>The Clustering Method</th>
<th>CCN</th>
<th>Density of CCN %</th>
<th>Epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOM</td>
<td>1210</td>
<td>26.30</td>
<td>20</td>
</tr>
<tr>
<td>K-Means</td>
<td>1083</td>
<td>23.54</td>
<td>20</td>
</tr>
<tr>
<td>Neural Gas</td>
<td>1050</td>
<td>22.82</td>
<td>20</td>
</tr>
<tr>
<td>GNG</td>
<td>967</td>
<td>21.02</td>
<td>5</td>
</tr>
<tr>
<td>RUFFNN</td>
<td>2731</td>
<td>59.36</td>
<td>1</td>
</tr>
</tbody>
</table>

5. Discussion and Conclusions

To overcome the problems of the low training speed and clustering accuracy, and high memory usage, the RUFFNN clustering method with one epoch training, data dimensionality reduction and controlling noise abilities was proposed. Table 5 shows time and memory complexities of the RUFFNN and some related UFFNN methods [4].

Table 5. Comparison of time complexities and memory complexities of the RUFFNN method with some related methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Time Complexity</th>
<th>Memory Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-means</td>
<td>$O(c.n.m)$</td>
<td>$O((n+k).m.s.o)$</td>
</tr>
<tr>
<td>NG</td>
<td>$O(c.n.m)$</td>
<td>$O(c.n,m,s_o)$</td>
</tr>
<tr>
<td>GNG</td>
<td>$O(c.n.m)$</td>
<td>$O(c.n,m,s_o)$</td>
</tr>
<tr>
<td>SOM</td>
<td>$O(n.m)$</td>
<td>$O(n,m,s_o)$</td>
</tr>
<tr>
<td>RUFFNN</td>
<td>$O(n.m)$</td>
<td>$O(n,m,s_o)$</td>
</tr>
</tbody>
</table>

The RUFFNN is a linear clustering method has the time and memory complexities of $O(n.m)$ and $O(n.m.s_o)$, respectively. The parameters $c$, $k$, $n$, $m$, $s_o$ are the number of epochs, clusters, nodes, attributes and size of each attribute. The experimental results showed the superior outcomes of the RUFFNN method. For future work, an online dynamic RUFFNN is suggested by improving the RUFFNN method.

References


Roya Asadi is a PhD of artificial intelligence (neural network) in computer science from the University of Malaya. Her interests include Artificial Neural Network modeling, Medical Informatics, Machine Learning, Data Mining, Image Processing.

Mitra Asadi is a PhD candidate of Entrepreneurship Technology in Islamic Azad University- Central Tehran Branch, Iran. Her interests include: English Language Translation, Entrepreneurship Technology, and Management.

Shokoofeh Asadi is Master of Agricultural Management Engineering from University of Science and research in Tehran- Iran. Her interests include: English Language Translation, Biological and Agricultural engineering, Management and leadership.