COMPARATIVE ANALYSIS OF THE TWO FUZZY NEURAL SYSTEMS ANFIS AND EFuNN FOR THE CLASSIFICATION OF HANDWRITTEN DIGITS

T Murali, N. Sriskanthan, Geok See Ng
School of Computer Engineering, Nanyang Technological University, Blk N4 #2A-32, Nanyang Avenue, Singapore 639798

ABSTRACT

Handwritten character recognition is an area with many applications. Over the last decade much research has gone into algorithms to develop systems, which accurately convert images of handwriting to text. At the same time, neuro-fuzzy classification models have been researched and proven to solve complex problems. In this paper, two popular models, Adaptive Neuro-Fuzzy Inference System (ANFIS) and Evolving Fuzzy Neural Network (EFuNN) are investigated. The paper will show how these two models perform in handwritten digits classification

1. INTRODUCTION

In the past few years, numerous methods have been applied to the task of handwritten character recognition. Many of these methods have involved advanced image processing and feature extraction algorithms to ensure high accuracy rates. In this paper, the author processes images, digits and uses neuro-fuzzy methodology for classification. Neuro-fuzzy computing is a popular framework for solving complex problems. It is the fusion of Artificial Neural Networks and Fuzzy Inference Systems. The author uses two popular modeling techniques, Adaptive Neuro-Fuzzy Inference System (ANFIS) and Evolving Fuzzy Neural Network (EFuNN). Both techniques whilst having similar architecture have different learning algorithms and can perform differently to a set of data. Thus the performance of these models is tested on the application of handwritten digit classification.

In following section, image processing and feature extraction are discussed. Sections 3 and 4 are the studies of ANFIS and EFuNN. Experiments and results comparing the two models on handwritten digits are described in Section 5. Section 6 concludes the comparison of the two models for handwritten digits.

2. IMAGE PROCESSING AND FEATURE EXTRACTION

Handwritten images have to go through stages of image processing before useful information can be extracted for classification purposes. Two image-processing procedures, edge enhancement and thinning are performed on the images to enable edge detection of the images. Edge detection reduces the amount of data that is needed to compute in the feature extraction process. After edge detection, feature extraction is performed. Feature extraction is an important task in the first step of the character recognition. Devijver and Kittler [1] defined feature extraction as the problem of “extracting from the raw data the information which is most relevant for classification purposes, in the sense of minimising the within-class pattern variability while enhancing the between-class pattern variability. KL transform is used as the feature extractor. The KL transform is a linear transform and corresponds to the projection of images onto the orthonormal eigenvectors of the covariance matrix of a large number of prototype images. The prototype images are representative of the types of images desired to be recognised by the recognition system, in this case, character images. The production of this transform is also known as principal component analysis [2, 3]. The KL transform requires the computation of the covariance matrix and is followed by its diagonalisation to produce the eigenvectors [4]. The resulting eigenvectors can be used as basis vectors for feature extraction. The first 10 elements are used as they are seen as the most significant from observation. The following sections discusses the neuro-fuzzy methodologies of interest.

3. ADAPTIVE-NEURO FUZZY INFERENCE SYSTEM

Adaptive-Neuro Fuzzy Inference System (ANFIS) is a five-layer Takagi-Sugeno fuzzy inference system [5, 6].

Assuming a Takagi-Sugeno type of fuzzy system having the following rule base.

1. If \( x \) is \( A_1 \) and \( y \) is \( B_1 \), then \( f_i = p_1 x + q_1 y + r_i \)
2. If \( x \) is \( A_2 \) and \( y \) is \( B_2 \), then \( f_i = p_2 x + q_2 y + r_i \)

where \( x \) and \( y \) are the input variables; \( A_1, A_2, B_1 \) and \( B_2 \) are the membership functions; \( f_i \) is the output; and \( p_1, q_1, r_i \) and \( r_i \) are the consequent parameters.

Let the membership functions of fuzzy sets \( A_i \) and \( B_i \) be \( \mu A_i \) and \( \mu B_i \) respectively.

Figure 1 shows the architecture of ANFIS. The Layer 1 is for fuzzification of the input variables. In the Layer 2, T-norm operators are used to compute the rule antecedent part.

\[
\mu f_i = \mu A_i (x) \mu B_i (y), \quad i = 1, 2
\]
The Layer 1 takes the input variables. At the Layer 2, each input variable is represented by a group of spatially arranged neurons to represent fuzzy quantisation. The task of this layer is to transfer the input values into membership degrees to which they belong to a membership function (MF). Thus, if an input variable fails to belong to any existing MF to a degree greater than a membership threshold, new neurons can be evolved.

Layer 3 normalizes the rule strengths
\[ \hat{w}_i = \frac{w_1}{w_1 + w_2} \]

(2)

Layer 4 is where the consequent parameters of the rule are determined.

Layer 5 computes the overall input as the summation of all incoming signals.
\[ f' = \hat{w}_1 f_1 + \hat{w}_2 f_2 \]

where \( f_1 \) and \( f_2 \) are the output of Layer 4.

\[ \hat{w}_1 = x \quad \text{and} \quad \hat{w}_2 = y \]

(4)

where \( x \) is current fuzzy input vector and \( y \) is its corresponding output vector. \( \hat{w}_1 \) is adjusted through supervised learning based on the output error and \( \hat{w}_2 \) is adjusted based on similarity on the cluster.

The Layer 3 contains rule nodes that evolve through learning. The rule nodes represent clusters of input-output data associations. Each rule node \( r \) is defined by two connection weights \( w_1 \) and \( w_2 \)

\[ w_1 = x \quad \text{and} \quad w_2 = y \]

The Layer 4 represents the fuzzy quantisation of the output variables similar to Layer 2. Using the weighted sum input function and a saturated linear activation function, the degree to which an output vector belongs to an output MF is calculated. At the Layer 5, a linear activation function is used to calculate the output variables (defuzzification)

The learning algorithm of EFuNN mainly consists of following steps: network initiation, inputs feeding forward, connections updating, parameters tuning, node aggregation, node pruning, and rule extraction.

5. EXPERIMENT ON HANDWRITTEN DIGITS CLASSIFICATION

In this paper, handwritten digits are classified using the earlier discussed neuro-fuzzy modeling techniques of EFuNN and ANFIS. The experiments are to determine the performance of EFuNN and ANFIS based on the following criteria.

1. Recall capability
2. Adaptability
3. Timing
4. Memory Requirement

The data set is a 3000x10 matrix. This data set is the result of image processing and feature extraction discussed in Section 2. Each row consists of 10 training values for a specific digit. These 10 values are obtained from the KL transform for a particular handwritten digit. Both systems are trained at 1 epoch, 25 epochs and 50 epochs. The choice of 1 epoch is made because it represents the initial stage of EFuNN learning when the system is initialised with maximum number of rule nodes. Thus it is expected that the accuracy would be proportionately high. 50 epochs represents multiple iterations of learning and thus due to the evolving nature of EFuNN, it is expected that the number of rule nodes will be adjusted and be lesser than that of 1 epoch. It would also be useful comparing the accuracy of classification at this stage with at 1 epoch. Training at 25 epochs is added to monitor the system performance between 1 epoch and 50 epochs.

The following abbreviations are used in the discussions in the following sections:
ANFIS-1: ANFIS trained with 1 epoch
ANFIS-25: ANFIS trained with 25 epochs
ANFIS-50: ANFIS trained with 50 epochs
EFuNN-1: EFuNN trained with 1 epoch
EFuNN-25: EFuNN trained with 25 epochs
EFuNN-50: EFuNN trained with 50 epochs

5.1 Recall Capability

Recall capability refers to the system’s ability to classify accurately when it is tested with the same set of data it was trained with. Accuracy is measured on a scale from 0 to 1. From Table 1, it is evident that the ANFIS systems at increasing epoch provide increasing recall capability. Figure 3 shows that the ANFIS-50 produces full recall capability for samples up to 500. From then, accuracy drops steadily to a minimum at 1500 samples. The accuracy then increases and stabilizes to about 0.88 till 3000 samples. The ANFIS-25 produces 1.0 (i.e. 100%) accuracy for samples of size to 200. Subsequently the accuracy drops steadily as in the case of ANFIS-50. The minimum of 0.8 occurs at 1500 samples. ANFIS-1 follows a similar trend to ANFIS-25 with its minimum occurring at 0.75

![Figure 3. Recall capability of ANFIS and EFuNN.](image)

The results in Table 1 indicate that EFuNN performs better at 1 epoch and 50 epochs as opposed to 25 epochs. It can be more clearly seen from Figure 3 that EFuNN fails to produce full recall ability at all samples. The accuracy of the system fluctuates with minimums occurring at 700 and 1750 samples. EFuNN-25 performs worse than EFuNN-50. However, EFuNN-1 produces 1.0 accuracy up to 200 samples. Accuracy fluctuates with a minimum of 0.95 occurring at 1500 samples. Both EfuuNN and ANFIS possess good recall capability for small sample sizes (i.e. 100-500). It is noted that their performance dips to a minimum at 1500-1750 samples before rising again. This phenomenon is consistent at all different number of epochs. The good performance at low sample values can be credited to the fact the quantity of test data is small. However as the sample data increases accuracy decreases and this is can be seen in Figure 3 for both the ANFIS and EFuNN systems. Thus it can be deduced that samples of 2000 and more are required at training before classification of digits can be made satisfactorily. For ANFIS, the performance is relatively proportional to epoch. The ANFIS-50 performs the best. However, it is more complicated for EFuNN. The performance of EFuNN-1 is better than EFuNN-25 and EFuNN-50. However from the results in Table 1 and Figure 3, it can be concluded that EFuNN at its optimum performs better than ANFIS.

5.2 Adaptability

Adaptability refers to a system’s capability in classifying when it is tested with data different from the training data. The adaptability test is carried out with 70% of the samples used for training and 30% used for testing. From Table 1, it is noticed that in contrast to the results in experiment 1, classification accuracy at low sample samples (500-1000) is low and increases as the sample size increases. This is because as the sample size increases, 70% of increasing sample size provides greater training data. From Figure 4, it is interesting to notice that the EFuNN system and the ANFIS system started at unique accuracy levels. It is noticed that the ANFIS systems show a common trend of accuracy at different number of epoch. The ANFIS-50 system produces the highest accuracy amongst the three different numbers of epoch. Thus it’s conclusive that for the ANFIS system, accuracy is relatively proportional to the number of epoch.

![Figure 4. Adaptability of ANFIS and EFuNN.](image)
As it is reflected on Table 1, the EFuNN system produces greater accuracy than the ANFIS system. However it’s harder to distinguish a pattern to differentiate the performance of the EFuNN system according to the number of epoch. EFuNN-1 and EFuNN-50 has better performance than EFuNN-25. At EFuNN-1, the system possesses the greatest number of rule nodes crediting to highest accuracy as opposed to EFuNN-25 and EFuNN-50. EFuNN-25 has lesser rule nodes than that of EFuNN-50 because of its lower accuracy than that of EFuNN-50. All three EFuNN systems’ accuracy fluctuates as the sample size data changes.

5.3 Timing

5.3.1 Training Time

On top of recall capability and adaptability, it is also very useful to have a system that has short training time. From Table 2, it’s very clear that the ANFIS system takes much longer to train when the number of epoch is greater than one. As the sample size increases, training time also increases. Thus, it can be concluded that training time is proportional to training data. For EFuNN, it can be seen from Table 2 that at higher number of epoch, not only is the training time much smaller, but fairly constant. The exception is EFuNN-1, which like ANFIS, has training time increase with increasing training data. This is because with a single epoch it takes a longer time for the system to train the data to meet the given error threshold. Therefore, with more training data, the training time is longer.

5.3.2 Classification Time

A trained system is tested with a set of test data. While it is important that a system is able to classify accurately based on its learning, it’s also useful to identify how fast it can reach a decision on classification. As it can be seen from Table 2, the EFuNN takes longer time in classification than that of ANFIS. Such longer time taken is due to the greater number of rule nodes it possesses as opposed to ANFIS. It can be deduced that the simulation times for EFuNN is consistently greater if the number of rule nodes is greater.

5.4 Memory Requirement

Since there is no direct method to calculate the memory being used at each of these systems, an estimate of the resource usage can be gauged from the number of rule nodes used in each system.

The number of nodes used by ANFIS does not change regardless of the number of epoch that is in contrast to EFuNN. Due to the evolving nature of EFuNN, the number of rule nodes change at each epoch. As it can be seen from Table 5, the number of rule nodes in EFuNN-1 increases with increasing number of samples. This is due to the learning algorithm of EFuNN. The system is initialised with the maximum number of rule nodes and the number of nodes is proportional to the training data. However at higher epoch number, the number of rule nodes stabilise regardless of training data size. By looking at Table 2, it is also noticed that EFuNN-1 is set to maximum number of rule nodes, at EFuNN-25 the number of rule nodes is in the 20s and at EFuNN-50 the number of rule nodes in the 40s. This is due to the evolving nature of the system where aggregation and pruning takes place at the various number of epoch. As it is seen earlier in Section 5.2, the numbers of rule nodes directly affect the classification results. Thus, at this point it is possible to conclude that as far as EFuNN is concerned a trade-off has to be made between memory consumption and accuracy. EFuNN-1 with its large number of rule nodes produces higher accuracy than EFuNN-50 that possess lesser nodes. Therefore, an EFuNN system with 50 epochs or greater is recommended as the system’s learning stabilises to an optimum number of rule nodes and its accuracy also comparable with EFuNN-1. Nevertheless, from the results, as far as memory consumption is concerned, ANFIS possess a relatively lower number of rule nodes that are similar at all number of epoch.

6. SUMMARY

Due to the lack of common framework, it remains difficult to compare the different neuro-fuzzy models conceptually and evaluate their performance comparatively. As mentioned earlier, as a guideline, a highly intelligent and efficient system should meet certain criteria. It should be adaptable to new inputs and be fast learning. The system should also take as little time as possible to accurately classify digits. It should also useful to have low RMSE. EFuNN posses better recall capability and adaptability than ANFIS based on the case study. EFuNN takes a lot lesser time to train but interestingly, ANFIS takes less classification time. This is due to the fact that EFuNN possesses more rule nodes than ANFIS. However it is noted that whilst ANFIS performed classification of the digits faster, its accuracy for the given test data is lower than EFuNN. This is logical since having more rule nodes enables classification of digit to a higher accuracy. In other words, EFuNN takes longer time but classifies data more accurately. However, ANFIS has lesser RMSE at both training and simulation. It also possesses lesser rule nodes which can be credited a lower memory consumption. Thus, from the investigation of the two models with respect to handwritten digits, it is found that a trade-off has to be made between accurate classification and fast training to quick classification and low memory consumption. However, in handwritten character recognition, due to the vast variability of the data, adaptability occupies a high precedence over memory consumption and quick classification. Thus, it’s recommended that EFuNN is more suitable for task of handwritten digit classification.
### 7. REFERENCES


