21

Designing a Pattern Recognition Neural Network with a Reject Output and Many Sets of Weights and Biases

Le Dung and Makoto Mizukawa
Shibaura Institute of Technology
Japan

1. Introduction

Most neural networks that have been designed to solve the problems of pattern recognition use a supervised training method with a training data set. This data set contains examples of input patterns together with the corresponding output results, and the neural network learns to infer the relationship between input patterns and output results through training.

In supervised training, we often try to find out a set of weights and biases for the neural network in order to classify all patterns in the training data set. In general, training with a larger training data set can reduce the recognizing error rate. However, it would be difficult to find out a good design of neural network that will be able to learn all patterns in a large training data set, because it usually contains some patterns that are difficult to classify. Even if network layers and neurons were added more, there are still some misclassified patterns after a long time training process. The number of these patterns will increase when the size of the training data set is enlarged. If the neural network has to recognize a pattern that approximates in shape to one of the misclassified patterns, the recognition result will be incorrect. Furthermore, if a new pattern is updated, which approximates in shape to one of the misclassified patterns in the old training data set, the neural network may not still classify it, and it will become a new misclassified; thus, the error rate will increase.

In this chapter, we introduce a new design of pattern recognition neural network that has a simple structure but is still able to classify almost all training patterns exactly. The neural network is designed with an especial output that is called "Reject output". With this output, a large training data set can be separated into some parts, and with a smaller number of patterns in each part, they can be classified by the neural network more easily using a distinct set of weights and biases. Additionally, we also design a training method with some phases, which helps the neural network with the reject output to find out not only one but many sets of weights and biases for classifying almost all the training patterns. All the sets of weights and biases have to be kept in the order that they have been received from the training process.

Moreover, the reject output is also used to control the updating process for new patterns more easily. With the reject output, the pattern recognition neural network can produce not only correct or incorrect results but also reject results; therefore, it can control the recognizing rejection and reduce the error rate.

Source: Pattern Recognition Techniques, Technology and Applications, Book edited by: Peng-Yeng Yin,
ISBN 978-953-7619-24-4, pp. 626, November 2008, I-Tech, Vienna, Austria

On the other hand, with this design, the size of the neural network can be reduced to be implemented on a hardware-based platform in order to make fast classifiers.

2. Neural network with a reject output and many sets of weights and biases

In this session, the idea of designing a reject output for a pattern recognition neural network will be presented, and then the reason why this neural network uses many sets of weight and biases will be also explained.

2.1 Problems with a large training data set

Most pattern recognition neural networks have been designed with a supervised training algorithm, which uses the training data set to adjust the network's weights and bias so as to minimize an error function, such as the mean squared error function (MSE) (Martin et al., 1996), and try to classify all patterns in the training data set. The neural network can be considered a transfer function that changes a pattern space into an output space, in which each pattern class is clustered in a separate area. Figure 1 shows an example of handwritten digit pattern recognition for the above principle. After training, the neural network will have a set of weights and biases that will be used to recognize the new patterns.

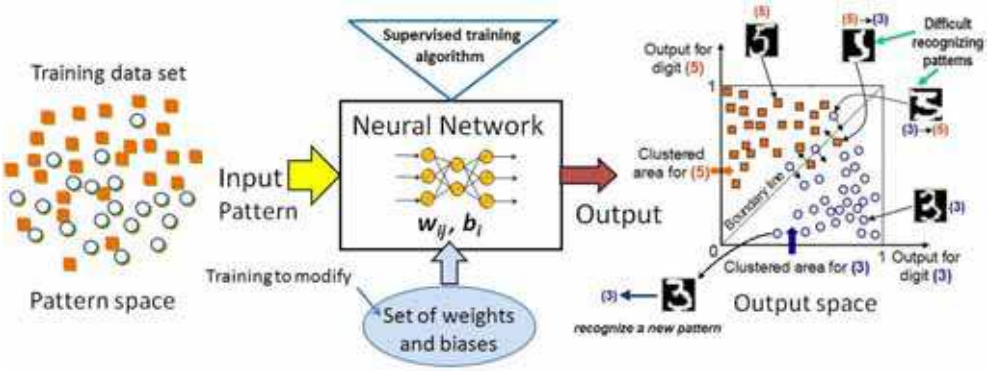


Fig. 1. Neural network is trained with a supervised training algorithm to cluster all training patterns from pattern space into output space.

In general, training neural network with a larger training data set can reduce the recognizing error rate, but there are some problems that we have to consider.

- Training neural network with a larger training data set, it requires more time to minimize the error function, especially when the data set contains some patterns that are difficult to classify correctly. Even though the neural network has been trained by many epochs (Martin et al., 1996), they are still clustered in a wrong area; hence, they are called the misclassified patterns (Gloger et al., 1997). These patterns keep some other patterns staying close to boundary line (Fig. 1); therefore, the error function reduces very slowly in training process.
- The number of the misclassified patterns will increase when the size of the training data set is enlarged. If the neural network has to recognize a pattern that approximates in shape to one of the above patterns, the recognition result will be incorrect.

- When some new patterns are updated to the training data set, the neural network must be trained with all old and new patterns, so it takes more time. Furthermore, if a new pattern approximates in shape to one of the misclassified patterns in the old data set, maybe it will become a new misclassified in the new data set.
- Adding more hidden layers and neurons to the neural network in some cases, it can classify more patterns in the large training data set. However, it is difficult to determine how many hidden layers and neurons we have to add. Moreover, with a large number of neurons and complex connections, the neural network definitely spends more time to bring out a recognition result.

2.2 The idea of reject output

To solve the above problems, we propose a new structure of pattern recognition neural network with an especial output that is called "Reject output", and build a training method corresponding to this structure. The name "Reject output" that means it is used to separate all difficult recognizing patterns (Fig.1) from the training data set. Hence, these patterns are called "Rejected patterns".

In order to explain the idea of the reject output straightforwardly, we will start with the single layer perceptron that was invented in 1957 by Frank Rosenblatt. The single layer perceptron with the perceptron learning rule is only capable to cluster linearly separable patterns (Frank, 1958). In the training pattern space, if there is not any hyperplane (or decision boundary) (Martin et al., 1996), which can separate all types of patterns perfectly, the training process of the perceptron is not guaranteed to converge. This was showed in a famous book entitled "Perceptrons" by Marvin Minsky and Seymour Papert in 1969 with the well-known problem that was called the exclusive-or (XOR) problem (Fig. 2).

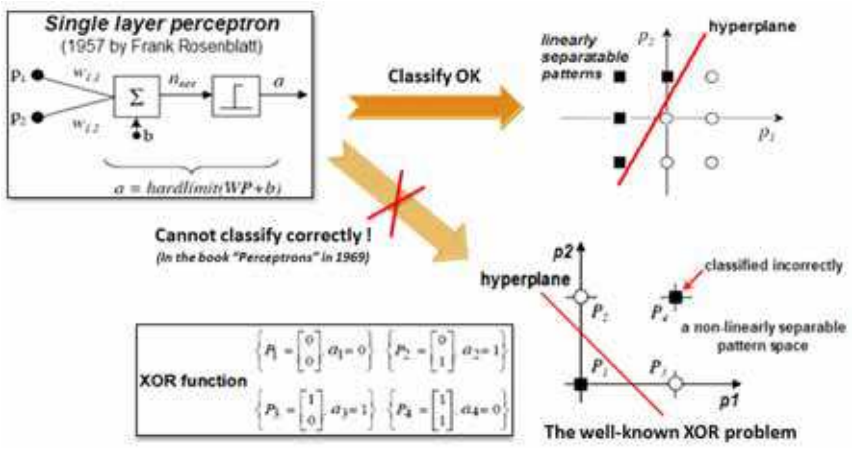


Fig. 2. The single layer perceptron and the XOR problem (Minsky & Papert, 1969).

Until the 1980s, the above limitation of the single layer perceptron was overcome with multilayer perceptron (Rumelhart et al., 1986) and back-propagation learning rule. Figure 3 illustrates the way that a multilayer perceptron with a hidden layer solves the XOR problem.

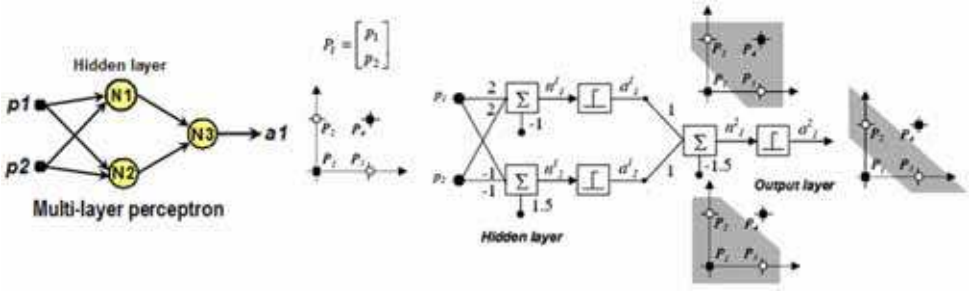
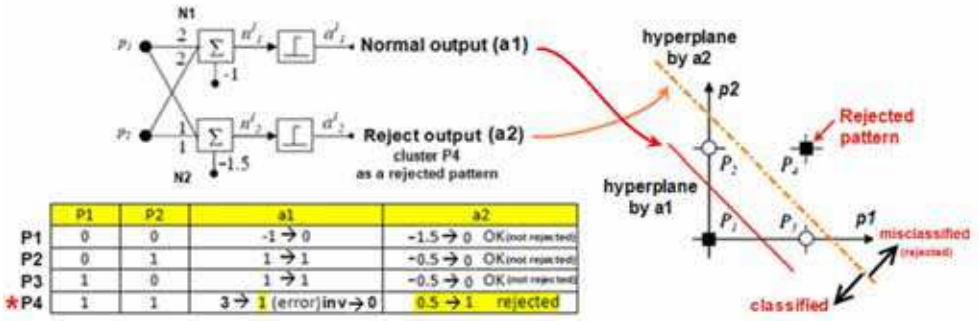


Fig. 3. A two-layer perceptron can solve the XOR problem (Martin et al., 1996).

We realized that adding a reject output to the single layer perceptron is also able to solve the XOR problem. In figure 2, the hyperplane separated the XOR patterns space into two areas for two types of the pattern, but the pattern – (P4) was classified incorrectly. Thus, (P4) is a misclassified pattern and (P1), (P2), (P3) are classified patterns, and now it can be considered as a new classifying problem with two classes: classified and misclassified. In this classifying problem, a hyperplane can be found out easily to separate perfectly misclassified pattern from classified patterns (Fig. 4). The reject output is added to the single layer perceptron to determine this hyperplane. Hence, (P4) will be masked as a rejected pattern and trained by the reject output. If the reject output (a2) is inactive, the normal output (a1) gives the correct result of the XOR function. If the reject output (a2) is active, the result given by the normal output (a1) is incorrect. In this two class problem, that means the inversion of this result (a1) is the correct result (Fig. 4).



	P1	P2	a1	a2
P1	0	0	-1 → 0	-1.5 → 0 OK(not rejected)
P2	0	1	1 → 1	-0.5 → 0 OK(not rejected)
P3	1	0	1 → 1	-0.5 → 0 OK(not rejected)
*P4	1	1	3 → 1 (error) inv → 0	0.5 → 1 rejected

Fig. 4. Using the reject output to solve the XOR problem.

In brief, the reject output plays a role to separate all misclassified patterns from classified patterns. It can also be said that the reject output is used to reject all misclassified patterns from the training data set. However, it cannot be said that the reject output always rejects perfectly all misclassified patterns. The example pattern space in figure 5 shows that the reject output (a2) cannot reject perfectly two misclassified patterns (PA3) (PA4) from all classified patterns that have already classified by the normal output (a1). The reject output tried to separate 2 misclassified patterns (PA3) (PA4) from the others, but it also rejected 2 classified patterns (PB3) (PB4). However, this problem leaded us to the idea to extend the training process in order to find more sets of weight and biases to classify almost all training patterns.

2.3 Neural network uses many sets of weights and biases

After training, the perceptron (Fig 5) with the reject output (a2) and a set of weights and biases can reject four patterns, such as (PA3) (PA4) (PB3) and (PB4). They are thus called rejected patterns. These rejected patterns include both types of training patterns, such as (A) and (B). And now, we can consider them as a new training data set to train the perceptron. With this new training data set, the perceptron can classify easily, and then the second set of weights and biases will be found out from this extended training process.

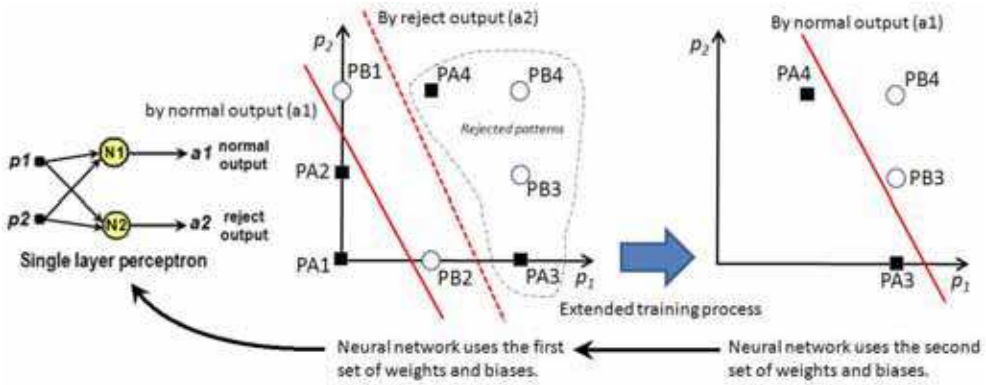


Fig. 5. Separate rejected patterns from training data set and then train them in the extended training process. The perceptron will use 2 sets of weights and biases to classify all patterns.

With a larger training data set, the number of the misclassified patterns will increase, that means the number of the rejected patterns that will be trained in the extended training process will also increase. If the normal output cannot classify all these patterns in the extended training process, the reject output will be used once more to separate misclassified patterns looks like the way in the previous training step. Therefore, we can receive more than 2 sets of weights and biases. The neural network will use all these sets of weights and biases to classify a new pattern in a fixed order. As a result, all the sets of weights and biases have to be kept in the order that we receive them from the training process. The training process can be divided into some phases.

2.4 Advantages and disadvantages

In principle, the pattern space in figure 2 can be clustered by 3 hyperplanes, it means that a multilayer perceptron with a hidden layer included 3 neurons can also cluster all patterns correctly. However, using the reject output and many sets of weights and biases still has some advantages in comparison with a multilayer perceptron without the reject output:

- With a simpler structure, the neural network will run faster. Moreover, this structure can be designed in parallel processing structure on a hardware-based platform; therefore the response time of the hardware-based neural network will be reduced.
- The neural network is designed with a reject output to separate the training data set into some parts, and with a smaller number of patterns in each part, they can be classified more easily.
- If a new training pattern is added to the training data set, maybe it would be a misclassified pattern when the neural network uses the first set of weights and biases,

but it is rejected by the reject output. In this case, we only have to train this pattern with other rejected patterns in the extended training process; thus, the neural network can learn this new pattern more easily.

- With the reject output, the reject rate of recognition can be controlled. We can increase the reject rate to reduce the error rate by changing a threshold at the reject output.

There are some disadvantages of this neural network:

- Because the training process has some phases, the training program is more complex.
- The data set of rejected patterns in the previous training process sometimes has a large number of patterns in the same class in comparison with the other classes; thus, training the neural network with this data set usually gives a not good set of weights and biases. Therefore, we have to select more patterns to add to this data set for training. How to select these patterns that is still a problem for studying.

3. Design of pattern recognition neural network with a reject output

The single layer perceptron with the reject output in the previous session is only a simple example to present basically our idea of neural network designed with a reject output and many sets of weights and biases. In order to interpret in detail the problems of this chapter, a design of neural network for recognizing handwritten digits patterns will be chosen as an illustrative example, because the handwritten digits recognition that is a typical application of pattern recognition neural network (Le & Mizukawa, 2006).

3.1 Structure of the pattern recognition neural network

There are many types of neural network that can be used for pattern recognition (Bishop et al., 1986). The convolutional neural network (CNN) is a famous type of pattern recognition neural network that has been successfully applied to handwritten character recognition (LeCun et al., 1995). The CNNs are designed to recognize visual patterns directly from pixel images with minimal pre-processing (LeCun et al., 1998). Thus, the convolutional structure was chosen for the design of pattern recognition neural network in this chapter. However, we intentionally designed a small and simple CNN, because we want to prove our small CNN can still classify almost all patterns in a large training data set by using the reject output and many sets of weights and biases. That is the main goal of this chapter.

Figure 6 illustrates our CNN designed for recognizing the handwritten digit pattern 16x16 pixels. Therefore, the input layer has 256 neurons arrange in a matrix 16x16. The CNN has only one convolutional layer and one sub-sampling (LeCun et al., 1998). The convolutional layer (C1) has only 2 non-symmetric feature maps of size 7x7 and 5x5. Each neuron in each feature map connects to the input layer in a matrix 4x4 neighbourhood type (LeCun et al., 1998). Moreover, this neighbourhood type is design with two unit overlap (Patrice et al., 2003) for the 7x7 feature map and one unit overlap for the 5x5 feature map. The sub-sampling layer (S1) has 2 feature maps of size 5x5 and 2 feature maps 3x3 using 3x3 neighbourhood type to connect to (C1) with two unit overlap. The neural networks for handwritten digits recognition have almost only 10 outputs corresponding to 10 digits (from 0 to 9); thus, our CNN also has 10 outputs that we called 10 normal outputs.

With this design, our CNN can be considered as a small CNN. We assert that it would be difficult to classify all patterns in a large training data set by the above CNN (without the reject output structure). In fact, we have already tried to train our CNN (without the reject

output) with a training data set of 5000 patterns. After more 500 epochs, there are still 267 misclassified patterns. Adding more feature maps to the convolutional layer and sub-sampling layer will help the CNN to classify more patterns in the training data set. However, the size of the CNN will increase considerably and there are still some misclassified patterns. The consideration leads us to design the reject output for our CNN to cluster all misclassified patterns, and then they are extracted from the training data set in order to set up a new training data set for classifying in the extended process (Fig. 6).

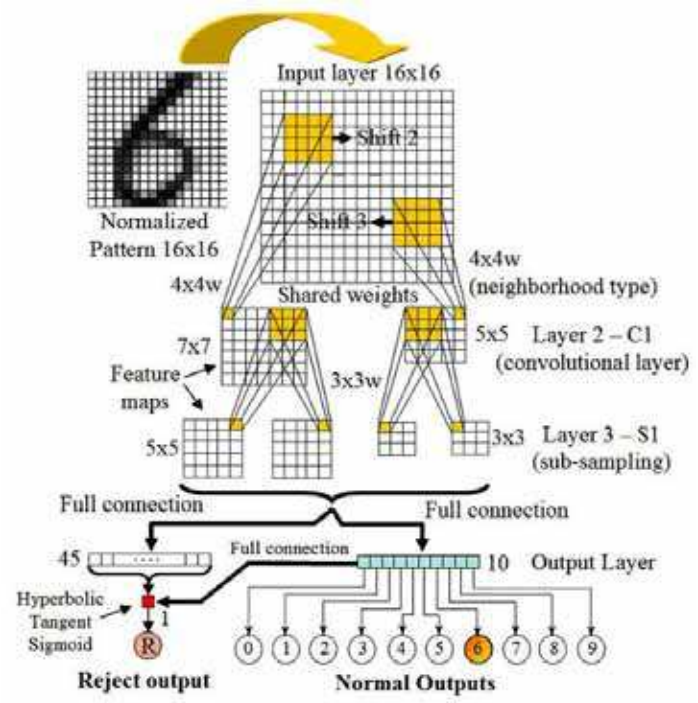


Fig. 6. A small convolutional neural network is designed with the reject output.

3.2 Design of the reject output

With the role of the reject output to classify all rejected patterns, which included all misclassified patterns and some classified patterns, the design of the reject output will be considered as a design of neural network. Therefore, we added to the CNN not only one neuron for the reject output but also a hidden layer before the reject output (Fig. 6). The hidden layer has 45 neurons corresponding to 45 areas between any two of ten directions in the patterns space. In fact, we have already tried to use 10, 25, 45, 55 and 100 neurons for this hidden layer and finally the hidden layer with 45 neurons is the best selection for the ability and speed of classification. All neurons of this hidden layer are fully connected to the sub-sampling layer (S1) of CNN. Especially, the reject output also connects to 10 normal outputs, because the values of 10 normal outputs are considered as important data for the rejection. Although 46 neurons are added to the CNN, it is still a small CNN in comparison with other CNN (LeCun et al., 1998) (Patrice et al., 2003). The activation function of the reject

output neuron is Hyperbolic Tangent Sigmoid that differs from 10 normal outputs with Log-Sigmoid function. Therefore, the value of the reject output is in range between -1 and +1.

$$\text{Reject output value} = a = \frac{e^{n_{net}} - e^{n_{net}}}{e^{n_{net}} + e^{n_{net}}} \quad (1)$$

a> threshold → the input pattern is rejected

a≤ threshold → the input pattern is not rejected

The threshold value should be determined for the reject output in order to decide between rejected pattern and un-rejected pattern. We will discuss this problem in section 4.

4. The training method with the reject output

Our training method is designed for a training process with many phases. The number of phases depends on the training data set that is used in the training process. If the training data set is not so large, the training process is often performed in four basic phases. With a larger training data set, the training process needs more extend phases. On the other hand, the training method is also concerned with the case when there are some new patterns that should be update to the training data set. For training the CNN by minimizing the mean squared error function (MSE), we have chosen the back-propagation algorithm (Bishop et al., 1986), which is perhaps the most widely used training algorithm for multilayer feedforward networks (Martin et al., 1996).

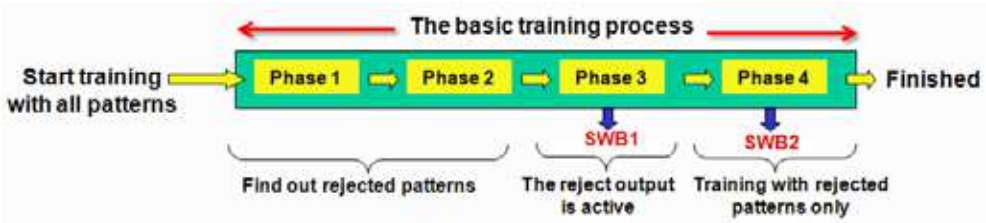


Fig. 7. The basic training process with four basic phases.

4.1 Four basic phases in the training process

- **The first phase**: the neural network is trained with all patterns in the training data set. The reject output is not active, that means the reject output neuron and 45 neurons in the hidden layer is not connected to the neural network. Back-propagation algorithm is used for 10 normal outputs. We use online-training method with 10 learning rates that are distributed for 10 outputs corresponding to their error values (Le & Mizukawa, 2006). In this phase, the biggest learning rate was used that is 0.5. This method makes the MSE to reduce faster. Our training software always tracks the decrease of MSE and the current total of misclassified patterns. Until two values reduce very slowly, we will switch to the second phase. After this phase, if there are some misclassified patterns, they will be marked as rejected patterns.
- **The second phase:** The neural network is still trained with all patterns but there is a small change. After the first phase, all misclassified patterns have already marked as rejected patterns, and the remains of training data set are classified patterns. To

separate all misclassified patterns from the classified patterns in the next phase more easily, in the second phase we train the neural network with all classified patterns in normal manner, but train all outputs with zero value for all rejected patterns. This training manner can push all rejected patterns toward the root (Fig. 8); thus, they will be clustered more easily in the third phase. However, in this phase, the reject output is still not active. The MSE usually continues to reduce in this phase, and we can increase the learning rate. Until MSE attains a low value, we will switch to the third phase. However, before doing the third phase, we should check all the rejected patterns and mark the rejected patterns again. In fact, some of the rejected patterns are classified correctly after the second phase; thus, they should be marked as the classified patterns.

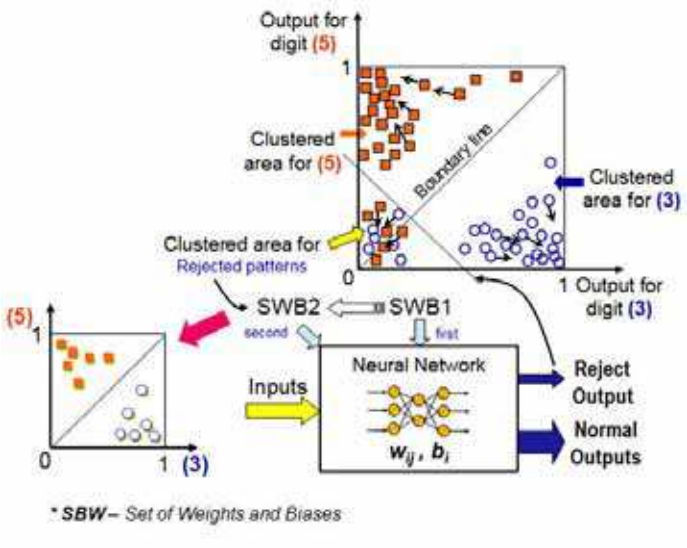


Fig. 8. Training the neural network with rejected patterns.

- **The third phase:** All patterns are used for training, and the reject output is active now to cluster all rejected patterns that were marked in the second phase. That is the most important phase in our training method.

In this phase, when a training pattern is placed into the input layer, the training program has already known the pattern is one of rejected or un-rejected pattern that correspond with misclassified or classified pattern in the second phase. If this pattern is one of rejected patterns, the neural network will be trained with value 0 for all ten normal outputs and value 1 for the reject output. If the pattern is an un-rejected pattern (or classified pattern), it will be classified by the corresponding normal output and the reject output is trained with value -1. In this manner, we want to continue clustering all rejected patterns in the area that is near the root point (Fig. 8).

We realize that the number of the rejected patterns is always smaller than the number of the un-rejected pattern, and if the neural network was trained with all rejected and un-rejected patterns in the original order of them in the training data set, the training process would take a long time to converge. Thus, the rejected patterns and the un-rejected patterns should be placed one after the other into the input layer of the neural network for training. The

training software can do it very easy. As a result, the rejected patterns are used in rotation in each training epoch; thus, they usually are clustered by the reject output faster than the un-rejected patterns.

The reject output value is always in range from -1 to +1. The neural network uses the reject output to cluster for the rejected patterns with value 1 and for the un-rejected patterns with value -1. Therefore, a threshold value R has been determined for the reject output to separate all the rejected patterns from the un-rejected patterns (Fig. 9). That means the minimum value α of all the values of the reject output corresponding to all the rejected patterns must be higher than R, and the maximum value β of all the values of the reject output corresponding to all the un-rejected patterns must be lower than or equal R. The training software can track the α and β in this phase, and the neural network should be trained until $\alpha > \beta$. At that time, the reject output can separate all the rejected patterns from the un-rejected patterns by the threshold $R = \alpha$.

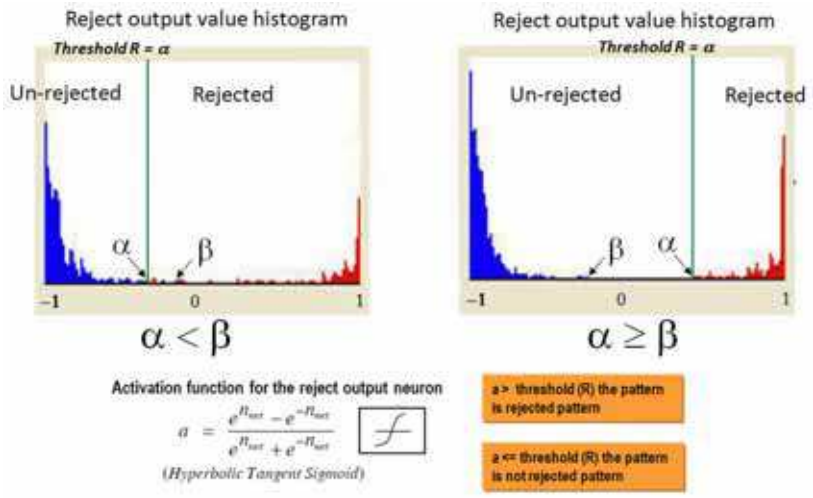


Fig. 9. Determine the threshold (R) for the reject output.

However, we do not need to wait until $\alpha > \beta$, because if $\alpha < \beta$ and $R = \alpha$, all the rejected patterns are still clustered by the reject output, although some un-rejected patterns are classified as rejected patterns (Fig. 9). It is no problem, because these patterns will be classified again in the next phase. Thus, if α is still smaller than β after hundreds of epochs, we should check all the training patterns and mark the rejected patterns again. If a pattern has already clustered by the rejected output, it is still marked as a rejected pattern. If a pattern is not classified correctly by the normal outputs, it should be marked as a rejected pattern. As the result, the number of the patterns that were marked as the rejected patterns may be changed in this phase. In brief, the rejected patterns and un-rejected patterns are classified flexibly. The threshold value R can be used to control the reject rate of the neural network. After this phase, we have the first set of weights and biases (SWB1) and the threshold value R for the reject output separated the training data set into two parts.

- **The fourth phase:** We must reset all weights and biases in order to start training the neural network with only the rejected patterns that have been classified by the reject

output in the third phase. The reject output is not active. If the number of the rejected patterns is not so large, the training process will converge in a short time. However, if the number of the rejected patterns is too small in comparison with the total of patterns in the training data set, or it includes a large number of patterns in the same class in comparison with the other classes, the set of weights and biases that we received in this phase is usually not good for recognizing. Therefore, we have to select more patterns to add to this training phase. In fact, we have had to use a smaller threshold value R to reject provisionally some patterns for this training phase. After this phase, the second set of weights and biases (SWB2) is determined. The training process is over and the neural network has two sets of weights and biases for recognizing.

4.2 Extended phases

If the number of the rejected pattern is large, the neural network might be not able to classify all of them in the fourth phase. Thus, the training process must be continued with some extended phases. The reject output will be used again to separate the set of the rejected patterns into 2 parts for classifying with the extend phases look like the third phase and the fourth phase (Fig. 10). In that way, we can separate a large training data set into more than two parts and use only one neural network with many sets of weights and biases to classify all patterns in each part. All the sets of weights and biases should be numbered in the order that we received them in the training process. If the size of neural network is reduced, the training process will require more extended phases, and the neural network will have many sets of weights and biases.

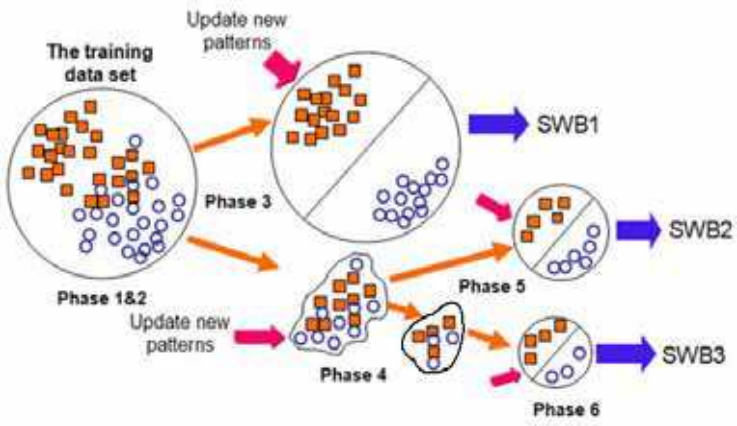


Fig. 10. With the reject output, a large training data set can be separated into more two parts to classify by using many sets of weights and biases (SWB).

4.3 Update new patterns to the training data set

First of all, we use the neural network to recognize all new patterns and check the recognition results. There are some cases of the recognizing result for each new pattern and the corresponding update method as follow:

- The new pattern is recognized incorrectly (misclassified) but the reject output does not reject this pattern. Thus, the pattern should be trained as an un-rejected pattern in the

third phase. After some epochs, if the pattern cannot be classified correctly, it will be marked as a rejected pattern for training with the reject output. After the reject output can be able to reject this pattern, it will be used to train the neural network in forth phase.

- In the case of the recognizing result is not correct and the reject output rejects this pattern, this pattern should be trained in the fourth phase.
- If the new pattern is recognized correctly but the reject output rejects this pattern, we should try to train this pattern as an un-rejected pattern in the third phase. If it takes a long time, the updating process should stop and return to the starting point, and then training this pattern as a rejected pattern in the fourth phase.
- In the case of the recognizing result is correct and the reject output does not rejects this pattern, the pattern could be trained as an un-rejected pattern in the third phase.
- We realize that the third phase and the fourth phase sometimes can perform simultaneously; thus, the time that we spend for updating new patterns is reduced.

5. Experimental result

The goal of our experiments is to prove a not big neural network with the reject output can be able to classify almost all training patterns and the recognizing ability of the neural network is improved.

The patterns that we use for training and testing are from the well-known MNIST database of handwritten digits (http:/ / yann.lecun.com/ exdb/ mnist). Our software extracts the patterns in the MNIST database (with 60000 training patterns) to build our own data sets, such as data sets of 600 patterns and 5000 patterns. We also built two testing data set with 200 patterns and 1000 patterns from the MNIST testing database (with 10000 testing patterns).

- **The first experiment**: The data set of 600 patterns was used to train the neural network. After 30 epochs of training in the first phase and second phase, 22 misclassified patterns were found out. They were masked as rejected patterns for training with the reject output in the third phase. In the third phase, 22 rejected patterns were separated easily from the training data set by the reject output with threshold R=0.76. With this threshold, these 22 patterns do not include any un-rejected patterns (we mean that is the case of $\alpha > \beta$). Then, these patterns were classified completely in the fourth phase. We received 2 sets of weights and biases (SWB1 and SWB2). Moreover, we also continuously tried to train the neural network in the first phase with hundreds of epochs but there were still 14 misclassified patterns. After this work, the neural network also has a set of weights and biases (SWB0) for testing. It is clear that the neural network with reject output and two 2 sets of weights and biases (SWB1 and SWB2) can classify all 600 training patterns correctly.

The recognizing ability of the neural network has been tested with 200 and 1000 testing patterns in some cases that were showed in Table 1.

From the testing results in Table I, we can see that the number of errors will be reduced if we use SWB1 and SWB2. If the neural network uses only SWB1 and the reject output is active, the number of errors is reduced more but we have to accept a number of rejected patterns, which may include some patterns that have been classified correctly by the normal outputs.

Testing data set	Use SWB0	Use SWB1 not use reject output	Use SWB1 with reject output	Use SWB1 and SWB2 with reject output
200 patterns	7 errors	11 errors	0 error 12 rejected	0 error
1000 patterns	166 errors	167 errors	138 errors 36 rejected	146 errors

Table 1. Testing results after training the neural network with 600 patterns.

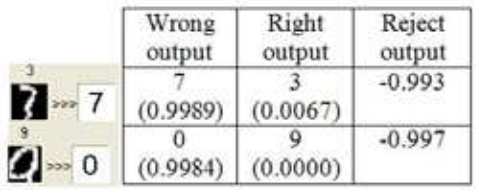
We also tested the neural network with some values of threshold R. Table 2 illustrates the testing results when we use the threshold R=0, R=0.5, R=0.76 and R=0.85 for the reject output. The threshold R=0.76 is equal the minimum value α of 22 values of the reject output corresponding to 22 rejected patterns that were found out in the third phase. From table 2, we can see that the result is best with R=α.

Test	R=0.76 (=α)	R=0	R=0.5	R=0.9
1000 patterns	146 errors 36 rejected 8 errors with SWB2	158 errors 94 rejected 53 errors with SWB2	150 errors 66 rejected 30 errors with SWB2	163 errors 5 rejected 0 errors with SWB2

Table 2. Testing the reject output with some threshold values

- **The second experiment**: we use the data set of 5000 patterns for training. The number of patterns that were marked as the rejected patterns for the fourth phases is 426, but that included 159 un-rejected patterns (we mean that is the case $\alpha<\beta$). With R=0.59, there were still 11 rejected patterns that the reject output cannot classified correctly. After checking these patterns, we realize that they look like other patterns very much; thus, it is difficult to cluster them by the rejected output. They should be rejected from the training data set because they are bad patterns (Fig. 11).

In the fourth phase, we tried to classify the above 421 patterns, but after hundreds of epochs, there were still 36 rejected patterns classified not correctly. Thus, we had to continue with two extended phases. After these extended phases, we found out 2 sets of weights and biases, such as SWB2 and SWB3. As a result, the neural network uses three sets of weights and biases to classify the 5000 training patterns with only 11 errors. The neural network was also tested with the 1000 testing patterns. Table 3 shows that the error rate will be reduced, if the neural network uses many SBWs with the reject output.



	Wrong output	Right output	Reject output
7	7 (0.9989)	3 (0.0067)	-0.993
0	0 (0.9984)	9 (0.0000)	-0.997

	Wrong output	Right output	Reject output
9	9 (0.9820)	5 (0.9764)	-1.000
5	5 (0.7182)	4 (0.5982)	-1.000

Fig. 11. Some bad patterns cannot be clustered by the reject output.

We also tried to train the neural network with 60000 patterns from the MNIST training database. That is really larger number for our small neural network. After the third phase,

with SWB1 and R=0 the reject output separated 12159 rejected patterns from the 60000 patterns, and there are 5903 un-rejected patterns that were clustered by the reject output. The neural network was trained with only these rejected patterns in the fourth phase and we received the SWB2, but there are still 2445 misclassified patterns.

Testing data set	Use SWB1 not use reject output	Use SWB1 and SWB2 with reject output	Use SWB1,SWB2 and SWB3 with reject output
1000 patterns	119 errors	103 errors 85 rejected	97 errors

Table 3. Testing results after training the neural network with 5000 patterns

Table 4 illustrates the testing results when the neural network tries to classify 10000 testing patterns from the MNIST testing database. We also see that the error rate 8.7 % will be reduced to 5.4 %, if the neural network uses SBW1 and SWB2 with the reject output.

Testing data set	Use only SWB1	Use SWB1 with reject output	Use SWB1,SWB2 with reject output
10000 patterns	870 errors	225 errors and 1699 rejected	540 errors

Table 4. Testing results after training the neural network with 60000 patterns

6. Future work

The main objective of our research is to design a smart vision sensor for the robots. This sensor will be designed with pattern recognition neural networks that require a small response time. The design of reject output for pattern recognition neural network in this chapter can reduce the size of pattern recognition neural network; thus, it can be applied to design our smart vision sensor in the near future. The structure of neural network with reject output has already opened some abilities to design the neural network in parallel processing structure that will be implemented on FPGA to make the smart sensor run faster (Fig. 12).

The manner that the neural network using the reject output to update a new pattern is one of studying directions to make the smart sensor can enhance its ability after it is commissioned.

7. Conclusion

Adding the reject output to the pattern recognition neural network is an approach to help the neural network can classify almost all patterns of a training data set by using many sets of weights and biases, even if the neural network is small. With a smaller number of neurons, we can implement the neural network on a hardware-based platform more easily and also reduce the response time of it. With the reject output the neural network can produce not only right or wrong results but also reject results. It is significant, if we design a neural network to help a robot to interact with people. The reject results can be accepted by the robot in this interaction process. If the neural network rejected a pattern, the robot would ask people to make the pattern again that looks like we talk "Pardon me".

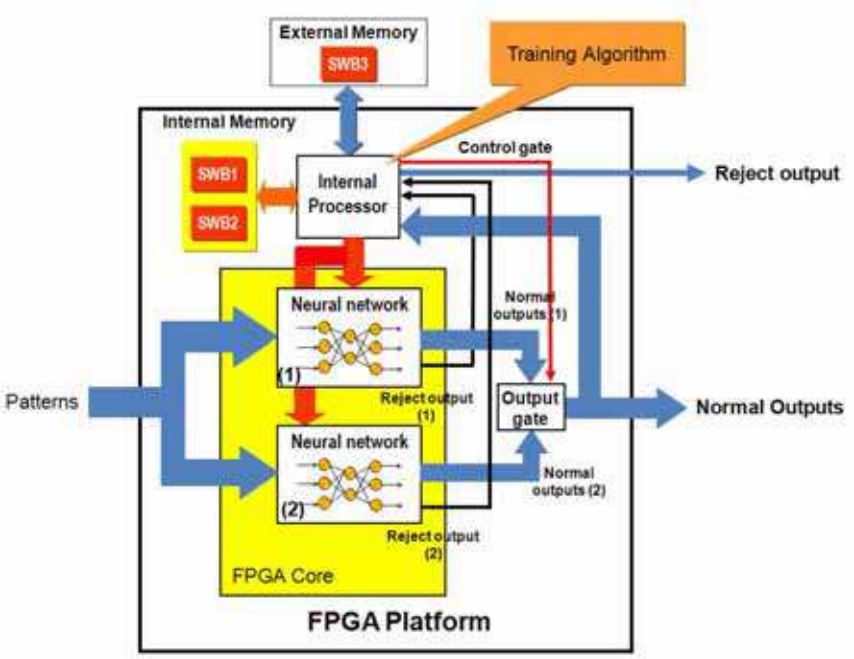


Fig. 12. Implement the neural network with the reject output on FPGA in parallel structure.

8. References

Bishop, C.M. (1995). *"Neural Networks for Pattern Recognition"*, Oxford University Press, USA

Frank Rosenblatt (1958), *"The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain"*, Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, pp. 386-408.

Gloger J. M., Kaltenmeier A., Mandler E., Andrews L.. (1997). "Reject Management in a Handwriting Recognition System", *Proceedings of the 4th International Conference on Document Analysis and Recognition,* pp: 556–559, ISBN: 0-8186-7898-4, IEEE Computer Society Washington, DC, USA

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, vol. 86, pp. 2278--2324, ISSN: 0018-9219, November 1998

LeCun, Y., L. D. Jackel, L. Bottou, A. Brunot, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Sackinger, P. Simard, and V. Vapnik. (1995). "Comparison of learning algorithms for handwritten digit recognition," *International Conference on Artificial Neural Networks, (F. Fogelman and P. Gallinari, eds.)*, pp. 53-60, Paris

Le Dung, Makoto Mizukawa. (2006). "Automatic handwritten postcode reading system using image processing and neural network technology", *Proceedings of SICE System Integration Division Conference (SI2006)*, pp. 930-931, Hokkaido, December 2006.

Martin T. Hagan, Howard B. Demuth, Mark Beale. (1996). *"Neural Network Design"*, Thomson Learning, ISBN: 981-240-485-6

Minsky, Marvin and Seymour Papert (1969), *"Perceptrons: An introduction to Computational Geometry"*, MIT Press.

Patrice Y. Simard, Dave Steinkraus, John C. Platt. (2003). "Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis", *International Conference on Document Analysis and Recognition (ICDAR)*, pp. 958-962, IEEE Computer Society, Los Alamitos, 2003

Rumelhart, David E., Hinton, Geoffrey E., Williams, Ronald J. (1986), *"Learning internal representations by error propagation"*, ISBN: 0-262-68053-X, MIT Press

Pattern Recognition Techniques, Technology and Applications

Edited by Peng-Yeng Yin

ISBN 978-953-7619-24-4
Hard cover, 626 pages
Publisher InTech
Published online 01, November, 2008
Published in print edition November, 2008

A wealth of advanced pattern recognition algorithms are emerging from the interdiscipline between technologies of effective visual features and the human-brain cognition process. Effective visual features are made possible through the rapid developments in appropriate sensor equipments, novel filter designs, and viable information processing architectures. While the understanding of human-brain cognition process broadens the way in which the computer can perform pattern recognition tasks. The present book is intended to collect representative researches around the globe focusing on low-level vision, filter design, features and image descriptors, data mining and analysis, and biologically inspired algorithms. The 27 chapters coved in this book disclose recent advances and new ideas in promoting the techniques, technology and applications of pattern recognition.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821