Systematic Treatment of Failures Using Multilayer Perceptrons

Fadzilah Siraj  
School of Information Technology  
Universiti Utara Malaysia  
06010 Sintok, Kedah  
Tel: 04-7003715, Fax: 04-7003715  
Email: fad173@webmail.uum.edu.my

Derek Partridge  
Computer Science Department  
University of Exeter  
Exeter Ex4 4PT, England

Abstract

This paper discusses the empirical evaluation of improving generalization performance of neural networks by systematic treatment of training and test failures. As a result of systematic treatment of failures, multilayer perceptron (MLP) discriminants were developed as discrimination techniques. The experiments presented in this paper illustrate the application of discrimination techniques using MLP discriminants to neural networks trained to solve supervised learning task such as the Launch Interceptor Condition 1 problem. The MLP discriminants were constructed from the training and test patterns. The first discriminant is known as the hard-to-learn and easy-to-learn discriminant whilst the second one is known as hard-to-compute and easy-to-compute discriminant. Further treatments were also applied to hard-to-learn (or hard-to-compute) patterns prior to training (or testing). The experimental results reveal that directed splitting or using MLP discriminant is an important strategy in improving generalization of the networks.

1.0 Introduction

Backpropagation networks with feedforward connections have by now established as highly competent classifiers (Waibel et al. 1989, Barnard and Casasent 1989, Burr 1988, Sarle 1999), but much remains to be discovered concerning the optimal design of such networks for particular applications. Issues such as the appropriate choice of features for input to the network (Barnard et al. 1991), the training methodology to be used (Jacobs 1988), and the best network topology (Obradovic and Yan 1990, Chester 1990) has all been identified, but complete satisfactory solutions have not been offered for any of these problems.

The accuracy (and hence reliability) of neural net implementations can be improved through a systematic treatment of the two failure cases: training failures and test failures. The paper addresses the basic problem of improving multilayer neural net implementations, i.e. increasing the generalization performance of the networks by developing methods for dealing with training and testing failure patterns.

The main objective of the experiments is to determine whether the Multilayer perceptron discriminant constructed from training or test patterns can be used to discriminate easy-to-learn (ETL) from hard-to-learn (HTL) patterns and in effect improves the generalization performance. The HTL pattern is defined based on the number of times that an input pattern fail on a number of networks or is known as the coincident failure (Partridge and Krzanowski 1997). Each pattern is considered as an unlearnt pattern (difficult pattern or referred as hard-to-learn or HTL if it fail on at least one network (for example), otherwise it will be considered as the learnt pattern (referred as easy-to-learn or ETL). If the discriminant was constructed from the training sets, the discriminant was referred to as the HTL/ETL discriminant. On the other hand, it is referred to as the HTC/ETC discriminant if it was constructed from the test sets. For HTC/ETC discriminant, two ways of defining the HTC were explored, namely as the pattern that fail in at least one net (referred as ONE HTC/ETC discriminant) and secondly as the pattern that fail in the majority of nets (MAJ HTC/ETC discriminant). The construction of such a set is first explained, then followed by experiments that will evaluate the accuracy of the MLP...
discriminant as well as improving the generalization performance.

To achieve the objective of the study, the first step is to divide the patterns into two separate groups. The first group of patterns will be used for training and testing. The easy and hard-to-learn patterns will be selected from these sets. The second group of patterns will also be divided into training and test sets that will be utilized to evaluate the performance of the MLP discriminant employed in the experiments.

The MLP discriminant was constructed using the easy and hard-to-learn patterns (or easy and hard-to-compute) selected from the patterns of the first group. Several combination of easy and hard-to-learn patterns are explored in order to identify the composition that produces the highest generalization.

Having determined the patterns, the next step is to perform experiments which should give some insight to the following plausible objectives:

1) To determine whether separating and modifying HTL patterns makes the patterns more learnable.
2) To determine whether learning improvement based on HTL/ETL discrimination leads to better computational reliability (i.e. a reduction in the number of wrongly computed results).

Once the HTL patterns were identified by the discriminant, further treatments were applied to these patterns. For modification purposes, normalization methods were employed. The first normalization method is to sum the squares of each parameter (i.e. \( x_1, y_1, x_2, y_2 \) and LENGTH), take the square root of the sum, and then divide each element by the norm (Bigus 1996, Cohn 1994 and Cohn 1974). This method is known as the Euclidean norm. A second method of normalization is simply by dividing each parameter by the parameter that has the largest value for a particular pattern (Bigus 1996).

### 2.0 Methodology

The Launch Interceptor problem has been used in a number of software engineering experiments concerning correctness and reliability (see Knight and Leveson 1986, Adams and Taha 1992). Partridge and Sharkey 1994, and Yates and Partridge 1995 have applied the problem to neural networks. This is a well-defined, abstract problem that has been chosen to study since it offers a distinct advantage of supplying numerous training and test patterns with unambiguous outcomes. The problem involves an anti-missile system, which is used to classify radar images as indicative of a hostile missile, or not. The input for the system represents radar images (specified as a sequence of \( xy \) coordinate points) together with 19 real-valued parameters and several small matrices which are used to control the interpretation of the radar images. The output is simply a decision Launch (when all 15 launch criteria are satisfied according to certain conditions) or No-Launch (when one or more of the 15 launch criteria is not satisfied by the input data. The various criteria upon which the decision depends are referred to as "launch interceptor conditions" (LIC’s). LIC1, the acronym from Launch Interceptor Condition 1, is a boolean function that is true if the Euclidean distance between two points is greater than the value of another parameter LENGTH, and false otherwise. The points are given as 2 pairs of \( x \) and \( y \) coordinates (each in the interval [0,1] to 6 decimal places) LENGTH is as single value in the same interval to the same precision. Therefore LIC1 takes 5 input values i.e. \((x_1, y_1), (x_2, y_2)\), LENGTH and returns the value true or false.

The results of preliminary studies show that training and testing performance can be improved by modification of a selected subset of patterns - ones with LENGTH <= 0.02, ones with LENGTH approximately equal to 'distance', those that fail to learn, and those that fail to test correctly. However, detailed problem-specific knowledge has been used to predict problematic patterns. In this study, the experiments were performed to determine whether it is possible to automatically predict (and to modify) these 'problematic' or HTL from ETL patterns using MLP discriminants. The initial objectives are to construct a 'representative' set of hard-to-learn (HTL) and learnable patterns, and to develop an MLP discriminant that can recognise these within a general set of patterns. In addition, the same idea with test sets, i.e. automatic prediction of hard-to-compute (HTC) patterns are also explored.

Three MLPs discriminants were constructed in the experiments. The first discriminant was constructed using the training patterns and it is known as the HTL/ETL discriminant. The second discriminant was constructed from the test patterns. From these patterns, the HTC was defined in two ways. The first definition of HTC refers to test pattern that fails in at least one network out of 27 networks. This discriminant is referred as ONE HTC/ETC discriminant. Another discriminant was constructed by defining HTC as a pattern that fails in the majority of networks. This discriminant is labelled as MAJ HTC/ETC discriminant.

The approach has been stimulated from the empirical studies conducted by Littlewood and Miller 1989 for software engineering research and Partridge and Sharkey 1992, Partridge 1994, Partridge and Griffith 1995, and Partridge and Yates 1996 from neural networks experimental research. In conjunction with these studies, MLP networks (Rumelhart and McClelland 1986) with 5 input units, 8 to 10 hidden units and 1 output unit (i.e. 5-8-1 to 5-10-1) are utilized in the experiments. Three training sets (Set1, Set2 and Set3) are used to investigate the effect of normalizing on the training and test patterns. Each training set is composed of 1000 random patterns trained
using an online backpropagation algorithm with a learning rate of 0.05 and momentum of 0.5. Sarle 1995 suggested that sample size of 1000 is large enough that overfitting is not a concern, so the results for 1000 training cases will provide a standard comparison for generalization results from small sample sizes. The weight seed number is varied from 1, 2 and 3. Each MLP network is trained to convergence (i.e. every pattern learned to a tolerance of 0.5 or 50,000 epochs for all training sets) whichever is first. Based on the training results, the patterns were classified as HTL or ETL patterns. As a result, some of these patterns were randomly selected to form discriminant set of patterns specifically for MLP discriminants. The performance of the networks was tested on 10,000 test patterns.

2.1 Constructing Discriminant Set

To construct a discriminant set, the previous 27 networks trained on three raw training sets were used. Each set was used to train 9 different networks (3 weight seeds x 3 hidden unit numbers), so there were nine attempts to learn each pattern of the 1,000 in each training set.

Let \( n \) represents the number of networks in which a pattern was not successfully learned, for example in training; \( n \) can vary from 1 to 9. If the patterns were not successfully learned in all the nine nets, then for an HTL pattern, then \( n \) is set to 9. Consequently, \( n \) can be set to 8,7,1 until \( \text{`enough'} \) HTL patterns are obtained. On the other hand, the ETL pattern is defined as the pattern that learned by all nine nets (in this case, the total number of network is 9). There are not many HTL patterns when \( n=9 \). Therefore \( n \) is set to a smaller number. The distribution of HTL patterns with respect to the number of pattern that was not learned in \( k \) out of 9 nets. Note that when \( n=5 \) for random training set Set1, the total of HTL patterns is equal to the sum of patterns from \( k=5 \) to \( k=9 \) (i.e. \( 0+0+0+0+1 = 1 \)). Thus, to get enough patterns for each class, \( n \) is set to 1 (i.e. the sum of patterns from \( k=1 \) to \( k=9 \)), and the composition of the random training sets for discriminate technique is therefore comprises of 2982 ETL and 18 HTL patterns.

Several possibilities will be explored in order to determine the training set composition that produced the highest average generalization. In conjunction with the chosen set the architecture for this particular set is determined, as well as the number of epochs that produces the highest average generalization. In effect, the \( \text{`chosen weight set'} \) is obtained and this is utilized in the next experiments as the HTL/ETL MLP discriminant.

In order to compare the discriminant techniques between MLP discriminants, the pattern sets used for discrimination purposes are also explored, viz:

1. A discriminant set is composed of the same number of ETL and HTL patterns. This set is referred as the EQUAL set.

2. Based on the results of the preliminary studies, it is known that the arrangement of training patterns for MLP training has an effect on the performance of the network. When the HTL and ETL subsets are constructed, there are not many HTL patterns. Intuitively, if there are too many ETL patterns, the networks may be trained to learn these patterns only. Therefore, to minimise the number of ETL patterns, we choose 3 of these patterns for each HTL pattern. The patterns are arranged in such a way that for every HTL pattern, it will be followed by 3 ETL patterns. These patterns are referred as the UNEQUAL set.

3. The same HTL is presented three times to each ETL. This set is referred as SAME HTL set.

2.2 Training and Testing

Having obtained, the MLP discriminants, several training and testing methods were explored, namely:

(1) The patterns are trained as MLPs without applying any treatment to the HTL patterns.

(2) Split and Separate

The HTL and ETL patterns are trained separately. The HTL are normalized before training or testing.

(3) The HTL and ETL patterns are trained separately but the HTL patterns are not treated before training or testing. This is known as the Split and Separate method without any treatments.
The experiments will be carried out in the same order as the objectives listed in section 1.0.

4.0 Results

The generalization of multilayer perceptrons without applying any treatments is 96.86%. Clearly, the generalization of all test methods exhibited in Table 1 are significantly higher than the MLPs without any treatment ($p = 0.0$). The Split and Separate method (without treatment) that uses HTC/ETC discriminant constructed from the UNEQUAL set achieved 100% generalization. The normalized method (99.66%) whose HTC/ETC discriminant was constructed using the EQUAL set achieved the second highest result. Like rational discriminants, the MAJ HTC/ETC discriminants obtained higher average generalization than ONE HTC/ETC discriminants. Although the split and separate method (without treatment) of ONE HTC/ETC constructed using the EQUAL set converges faster than the first normalized method of MAJ HTC/ETC, its generalization is 0.6% lower than latter. Hence the assumption that MAJ HTC/ETC discriminants produced higher average generalization is confirmed.

Results exhibited in Table 1 also indicates that the MAJ HTC/ETC discriminants achieved higher generalization than ONE HTL/ETL discriminants. Although untreated Split and Separate of ONE HTL/ETL constructed using the EQUAL set converges earlier than the first normalized method of MAJ HTC/ETC (600 versus 3389), the latter obtained 0.27% higher generalization than the first method. Thus, the generalization of the MAJ HTC/ETC discriminants is higher than ONE HTL/ETL discriminants.

Table 1: The random discriminant

<table>
<thead>
<tr>
<th>Type</th>
<th>Random</th>
<th>Disc. Set</th>
<th>Split and Separate Method</th>
<th>Test</th>
<th>HTC Set</th>
<th>ETC Set</th>
<th>Training sp.</th>
<th>$\chi^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAJ HTC/ETC</td>
<td>EQUAL</td>
<td>Normalized</td>
<td>Normalized</td>
<td>100.00</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1317.30</td>
</tr>
<tr>
<td>MAJ HTC/ETC</td>
<td>UNEQUAL</td>
<td>Normalized</td>
<td>Normalized</td>
<td>98.46</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1216.40</td>
</tr>
<tr>
<td>ONE HTC/ETC</td>
<td>EQUAL</td>
<td>Normalized</td>
<td>Normalized</td>
<td>95.96</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1301.25</td>
</tr>
<tr>
<td>CMA HTC/ETC</td>
<td>EQUAL</td>
<td>Normalized</td>
<td>Normalized</td>
<td>95.96</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1301.25</td>
</tr>
<tr>
<td>ONE HTC/ETC</td>
<td>UNEQUAL</td>
<td>Normalized</td>
<td>Normalized</td>
<td>93.10</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1247.39</td>
</tr>
<tr>
<td>CMA HTC/ETC</td>
<td>UNEQUAL</td>
<td>Normalized</td>
<td>Normalized</td>
<td>93.10</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1247.39</td>
</tr>
<tr>
<td>ONE HTL/ETL</td>
<td>EQUAL</td>
<td>Normalized</td>
<td>Normalized</td>
<td>93.10</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1247.39</td>
</tr>
<tr>
<td>ONE HTL/ETL</td>
<td>UNEQUAL</td>
<td>Normalized</td>
<td>Normalized</td>
<td>93.10</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1247.39</td>
</tr>
</tbody>
</table>

The $\chi^2$ analysis reveals that the MAJ HTC/ETC and ONE HTC/ETC discriminants which produces higher $\chi^2$ value for training patterns would achieve higher average training performance. As a result, the discriminant that obtained higher $\chi^2$ value for the test patterns would yield higher generalization performance. However, the ONE HTL/ETL discriminant that produces lower $\chi^2$ value for test patterns would yield higher average generalization performance.

5.0 Conclusion

The final findings from the experiments show that Split and Separate method without any treatment is one of the important training and testing method. Splitting method inevitably requires more resources and may also speed up the learning process. However, another question arises, is simply random splitting can improve generalization performance or is the directed splitting is more important? To answer this question, 2 experiments on random splitting methods were conducted on the same training and test sets, and the results are reported in Table 2. The results show that both random splitting methods affect the performance of the networks. In fact, the performance becomes worst than SRraw by at least 1.19%. Therefore the experimental results indicate that directed splitting is an important strategy in improving the generalization of the networks.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>50% to 50%</td>
<td>99.88</td>
<td>95.37</td>
<td>97.93</td>
<td>96.44</td>
</tr>
<tr>
<td>30% to 70%</td>
<td>99.82</td>
<td>95.45</td>
<td>97.46</td>
<td>96.67</td>
</tr>
</tbody>
</table>

Table 2: The training and test results using random splitting

References


