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/E*TC 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 prelimary 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 recognises 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 **MLP**s 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 definintion 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 `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 `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.

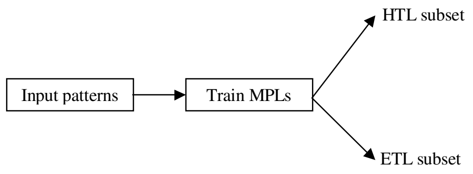

Figure 1: The procedure for obtaining HTL and ETL subsets from training patterns of ***SR**$_{raw}$*

(2) Split and Separate
The *HTL* and *ETL* patterns are trained separately. The *HTL* are normalized before training or testing.

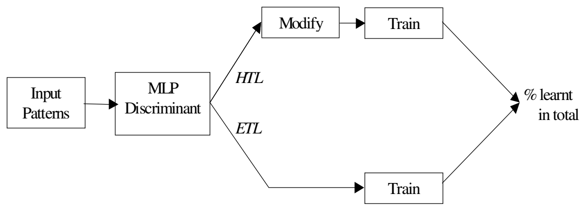

Figure 3: The procedure for obtaining the training performance for the Split and Separate method.

(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.

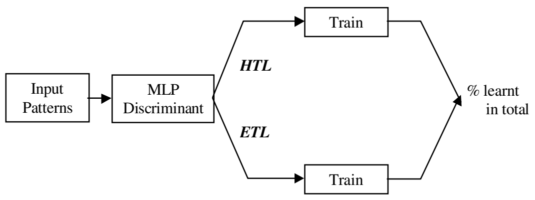

Figure 4: The procedure for obtaining the training perfomance by splitting the sets without treatment.

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.

Type	Random Disc. Set	Split and Separate Method	Gen.	*HTC* failure	*ETC* failure	Training ep.	χ^2
MAJ. HTC/ETC	*UNEQLAL*	*No treatment*	**100.00**	**0**	**0**	**3**	**1163.80**
MAJ. HTC/ETC	*EQUAL*	*Normalized 1*	**99.66**	**36**	**46**	**3388.89**	**444.16**
ONE HTC/ETC	*EQUAL*	*Normalized 2*	99.06	201	353	1962.967	1204.37
ONE HTC/ETC	*HTL1*	*Normalized*	98.86	166	446	2181.48	1019.59
ONE HTL/ETL	*EQUAL*	*No treatment*	**99.39**	**91**	**65**	**600**	**1330.62**
ONE HTL/ETL	*HTL1*	*No treatment*	98.46	194	369	16000	1626.92

Table 1: The random discriminant

The χ^2 analysis reveals that the *MAJ HTC/ETC* and *ONE HTC/ETC* discriminants which produces higher χ^2 value for training patterns would achieve higher average training performance. As a result, the discriminant that obtained higher χ^2 value for the test patterns would yield higher generalization performance. However, the *ONE HTL/ETL* discriminant that produces lower χ^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 $\boldsymbol{SR_{raw}}$ by at least 1.19%. Therefore the experimental results indicate that *directed* splitting is an important strategy in improving the generalization of the networks.

		Testing (in %)		
Split	Trainning (in %)	Min.	Max.	Gen.
50% to 50%	99.88	95.37	97.93	96.44
30% to 70%	99.82	95.45	97.46	96.67

Table 2: The training and test results using random splitting

References

Adams, J. and Taha, A. 1992. An experiment in software redundancy with diverse methodologies. In *Proceedings of the 25th Hawaii International Conference on System Sciences*, 83-90.

Barnard, E. and Casasent, D. 1989. A comparison between criterion functions for linear classifiers, with an application to neural nets. *IEEE Trans. Syst., Man, Cybern.* 19:1030-1041.

Barnard, E., Cole, R., Vea, M., and Alleva, F. 1991. Pitch detection with a neural-net classifier. *IEEE Trans. Signal Processing* 39:298-307.

Bigus, J. P. 1996. *Data Mining with Neural Networks.* New York, Mc-Graw Hill.

Burr, D. 1988. Experiments on neural net recognition of the spoken and written text. *IEEE Trans. Acoust, Speech, Signal Processing* ASSP-36:1162-1168.

Chester, D. 1990. Why two hidden layers are better than one. In *Proceedings of the International Joint Conferences on Neural Networks*, I-265-I-268, Washington, DC.

Cohn, P. 1974. *Algebra: Volume 1.* John Wiley and Sons.

Cohn, P. 1994. *Elements of Linear Algebra*. Chapman and Hall.

Jacobs, R. 1988. Increased rates of convergence through learning rate adaptation. *Neural Networks* 1(1):295-308.

Knight, J. and Leveson, N. 1986. An experimental evaluation of the assumption of independence in multiversion programming. *IEEE Trans. Software Engineering* 12(1):96-109.

Littlewood, B. and Miller, D. 1989. Conceptual modelling of coincident failures in multiversion software. *IEEE Transactions on Softe\ware Engineering* 15(12).

Obradovic, Z. and Yan, P. 1990. Small depth polynomial size neural networks. *Neural Computation* 2:402-404.

Partridge, D. and Griffith, N. 1995. Strategies for improving neural net generalization. Neural Computing and Applications 3:27-37.

Partridge, D. and Krzanowski, W. 1997. Distinct failure diversity in multiversion software. Technical report, 348, Department of Computer Science, Exeter University.

Partridge, D. and Sharkey, N. 1992. Neural networks as a software engineering technology. In Proceedings of 7th Knowledge-Based Software Engineering Conference, Sept. 20-23, McLean, VA, USA.

Partridge, D. and Sharkey, N. 1994. Neural computing for software reliability. *Expert Systems* 11(3):167-176.

Partridge, D. and Yates, W. 1995. Letter recognition using neural networks: a comparative study. Technical report, 334, Department of Computer Science, Exeter University.

Partridge, D. Yates, W. 1996. Engineering mutliversion neural-net systems. *Neural Computation* 8(4):869-893.

Rumelhart, D. and McClelland, J. 1986. Learning internal representations by error propagation. In Rumelhart, D., Hinton, G., and Williams, R., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, 318-362. MIT Press: Bradford Books.

Sarle, W. 1995. Stopped training and other remedies for overfitting. In *Proceedings of the 27th Symposium on the Interface*.

Sarle, W. 1999. Neural Network FAQ, part 3 of 7: Generalization, periodic posting to the Usenet Newsgroup comp.ai.neural-nets, URL: http://ftp.sas.com/pub/neural/FAQ.html.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. 1989. Phoneme recognition using time-delay neural networks. *IEEE Trans. Acoust, Speech, Signal Processing* 37:328-339.