IMPROVING NEURAL NETWORKS GENERALIZATION USING DISCRIMINANT TECHNIQUES

Fadzilah Siraj

School of Information Technology, University Utara Malaysia,
06010 Sintok, Kedah, Malaysia
Tel: 00-60-4-9284672, Email: fad173@uum.edu.my

Abstract

Neural networks, particularly Multilayer Pereceptrons (MLPs) have been found to be successful for various supervised learning tasks. Empirical studies have shown that the neural network is of powerful capabilities for pattern classification. Neural networks are non-linear models that classify based upon pattern recognition capabilities. During the learning process, neural network can build unique structures and select appropriate combinations of features that maximize the sensitivity of the classification decision. The classification decision depends on the input features. Hence this paper addresses the problem of improving neural networks implementations by developing methods for dealing with training and test failure patterns. The approach has been stimulated from the empirical studies from software engineering research (see [1]) and neural networks experimental research ([2], [3], [4]). The main objective of the experiments is to determine whether the multilayer perceptron discriminant constructed from training and 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 experiments presented in this paper illustrate the applic ation of discrimination techniques using MLP discriminants to neural network trained to solve supervised learning task such as the Launch Interceptor Condition 1 problem. The experimental results indicate that directed splitting or using MLP discriminant is an important strategy in improving generalization of the networks.

Keywords: Neural networks, Generalization, Backpropagation, Discriminant, Multilayer Perceotron

1. Introduction

Neural networks are a wide class of flexible nonlinear regression and discriminant models, data reduction models, and nonlinear dynamical systems ([5]). They are general functions approximators and they can therefore be trained to compute any desired functions including decision-making functions. Neural networks, particularly Multilayer Pereceptrons (MLPs) have been found to be successful for various supervised learning tasks. Empirical studies have shown that the neural network is of powerful capabilities for pattern classification. They are non-linear models that classify based upon pattern recognition capabilities. During the learning process, neural network can build unique structures and select appropriate combinations of features that maximize the sensitivity of the classification decision. The discrimination decision depends on the input features.

The common Linear Discriminant Analysis (LDA) is a well-known method for improving discrimination properties and compressing information in statistical pattern classification On the other hand, neural networks are inherently discriminative and yield higher classification accuracy than LDA ([6] and [7]).

The art of neural network design involves specifying three key elements, the neural network architecture, the input and output representations, and the training method. The architecture defines the connectivity of the processing units and their dynamics. The input and output representations encode the information (e.g. word, patterns) fed into and read from the net in terms of a pattern of neural activity. The training methods specifies how the weights are determined from the data.

The success or failure of the neural network is based on the appropriate selection of the above elements. This paper focuses on the input representations that are fed into the nets. 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 and test patterns can be used to discriminate easy-to-learn (ETL) from hard-to-learn (HTL) patterns and in effect improves the generalization performance.

2. Methodology

The Launch Interceptor problem has been used in a number of software engineering experiments concerning correctness and reliability (see [8], [9]). [10] and [11] 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**.

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.

2.1 Constructing Discriminant Set

The MLPs discriminants were constructed using the training and test patterns. The approach has been stimulated from the empirical studies conducted by [1] for software engineering research and [2], [12], [1] and [4] from neural networks experimental research. In conjunction with these research, voting strategy has been introduced when constructing the discriminant sets since voting algorithms have been shown to be very successful in improving the accuracy of certain classifiers ([13]).

The discriminant constructed from the training patterns is known as the HTL/ETL discriminant. On the other hand, the discriminants that were constructed form the test patterns are known as the HTC/ETC discriminants. In this study, the discriminant sets were constructed from the previous 27 networks trained on three raw training sets. Each raw training set was used to train 9 different networks (3 different weight seeds x 3 different hidden units), so there were nine attempts to learn each pattern of the 1,000 in each training set. A test set consisting 10,000 patterns was used to test the 27 networks

For the HTL/ETL discriminant, a pattern is chosen from the training set if the pattern was not successfully learned in several networks. In this case, let n represents the number of networks in which a pattern was not successfully learned. When n=5 (majority vote), not many patterns failed after training. Hence, n is set to 1 that means the patterns that was not successfully learned in at least one out of nine networks will be chosen to be included in the discriminant set. Out of 1,000 patterns, a total of 18 HTL patterns were identified and the rest of the patterns are considered as ETL patterns (2982). Since the percentage of ETL is very high compared to the HTL, several possibilities will be explored in order to determine the training set composition that produced the highest average generalization. The discriminant sets constructed for discrimination purposes are as follows:

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, 3 of these patterns are chosen randomly 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 *HTL1* set.

Similarly, the MLPs discriminants were constructed from the test patterns. Two HTC/ETC discriminants were constructed. For the first discriminant, n is set to 1, thus the ONE HTC/ETC discriminant is obtained. The second discriminant known as MAJ HTC/ETC discriminat was constructed by setting n = 5. Having defind the HTC and ETC patterns, the discriminant sets were constructed in the same manner as the training patterns.

2.2 Experiments

For the evaluation of the MLP discriminants, several experiments were carried out. Several training and testing methods were explored, namely:

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

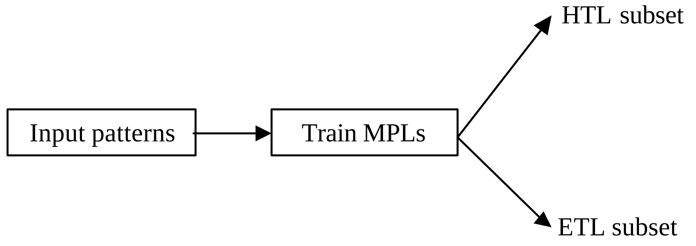

Fig. 1 The procedure for obtaining HTL and ETL subsets from training patterns of *MLP$_{raw}$*

(2) Split and Separate

The *HTL* and *ETL* patterns are trained separately. The *HTL* are modified by normalizing before training or testing. Two normalization techniques were explored. The first normalization method is to divide a particular attribute with the square root of sum of squares for all attributes for this pattern ([14], [15] and [16]). This approach is labeled as ***Normalized 1***. The second normalization method is to divide a particular attribute with the maximum value for a specific pattern (Bigus, 1996). This method is referred as ***Normalized 2***.

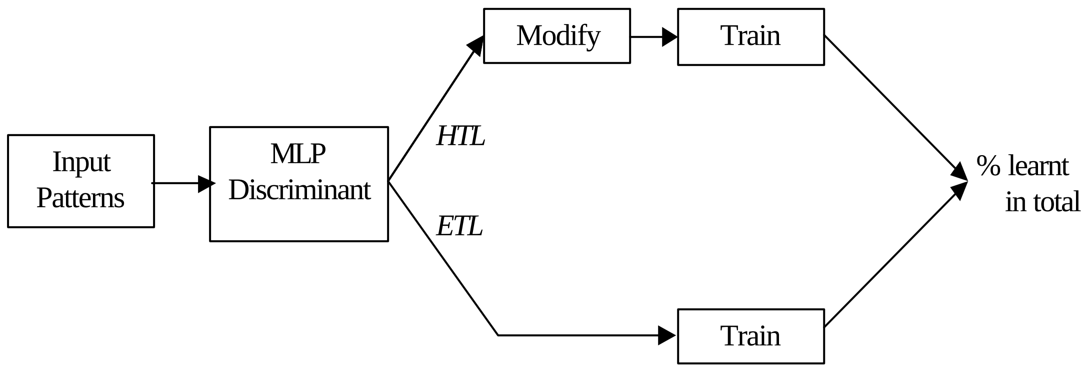

Fig. 2 The procedure for obtaining the training performance for the Split and Separate method.

(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.

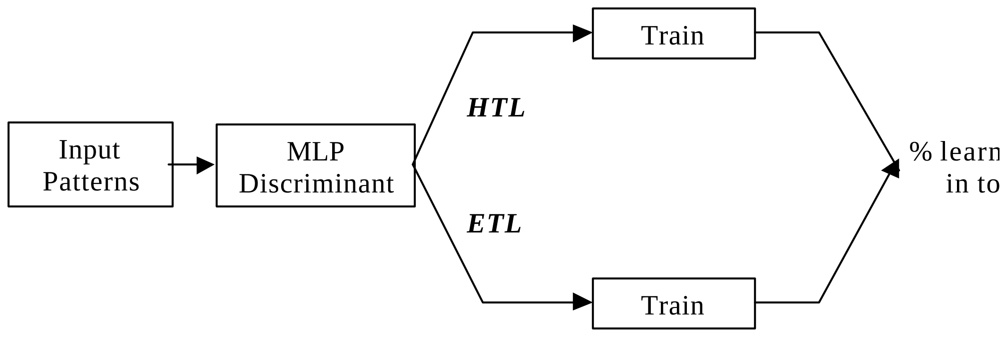

Fig. 3 The procedure for obtaining the training perfomance by splitting the sets without treatment.

3. Results

The average generalization of MLPs without any treatments or $\boldsymbol{MLP_{raw}}$ is 96.86%. This result is higher when compared with the results obtained using Linear Discriminant Analysis (90.4%). However, the average generalization for all test methods illustrated in Figure 4 are significantly higher than the results of $\boldsymbol{MLP_{raw}}$ ($p = 0.0$). The results also indicate that *HTC/ETC discriminants* defined using majority voting achieved higher average generalization results than the *ONE HTC/ETC* discriminants (see also [13]). The discriminants constructed the *UNEQUAL* set achieved 100% generalization. The normalized method whose *HTC/ETC* discriminant was constructed using the EQUAL set achieved the second highest result (99.66%). Although the ***split and separate*** method (without treatment) of *ONE HTC/ETC* constructed using the *EQUAL* set converges faster than the *MAJ HTC/ETC* constructed using the *EQUAL* set (1963 versus 3389 epochs), its generalization is 0.6% lower than the latter. Hence the assumption that *MAJ HTC/ETC* discriminants produced higher average generalization is confirmed.

Further observation on the average generalization results displayed in Figure 4 reveals 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.

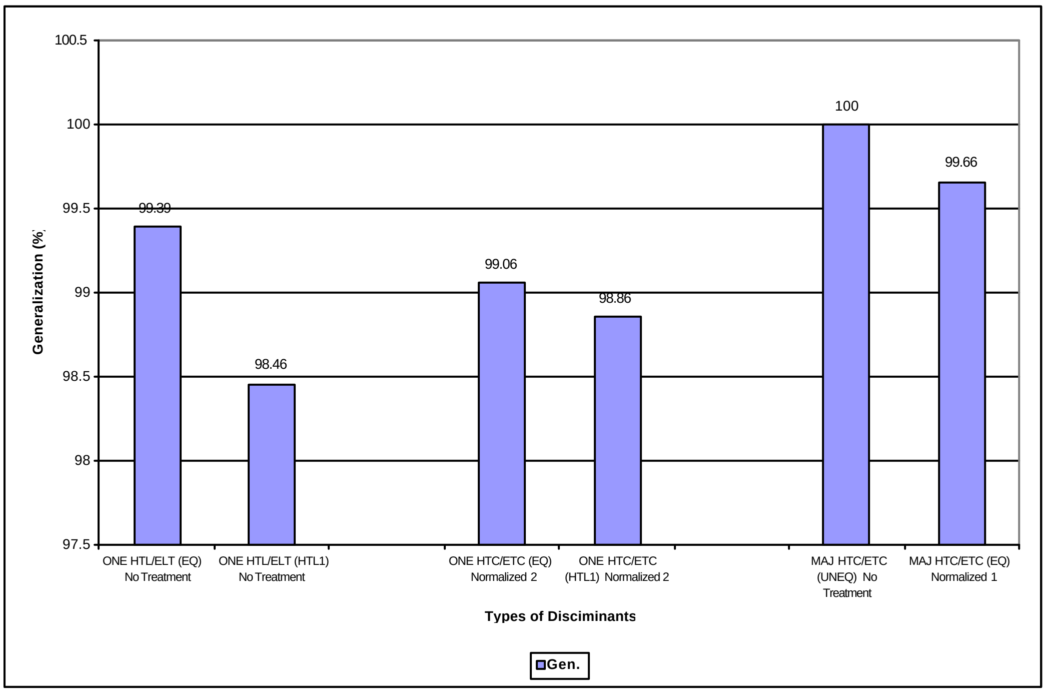

Fig. 4 The Generalization of MLP discriminants

4. 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 1. The results show that both random splitting methods affect the performance of the networks. In fact, the performance becomes worst than $\boldsymbol{MLP_{raw}}$ by at least 0.19%. Therefore the experimental results indicate that *directed* splitting is an important strategy in improving the generalization of the networks.

Table 1 The Generalization results using random splitting

Split	Generalization (%)
50% to 50%	**96.44**
30% to 70%	**96.67**

References

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

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

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

[4] Partridge, D. Yates, W.; Engineering mutliversion neural-net systems., Neural Computation, 8(4), pp. 869-893, 1996.

[5] Sarle, W. S.; Neural Networks and Statistical Models, Proceedings of the Nineteenth Annual SAS Users Group International Conference, April, 1994.

[6] Reichl, W., Harengel, S., Wolfertstetter and Ruske, G.; Neural Networks for Nonlinear Discriminant Analysis on Continous Speech Recognition, In the Proceedings of EUROSPEECH-95, Madrid, Spain, September, 2163-2166, 1995.

[7] Lerner, B., Guterman, H., Aladjem, M. and Dinstein A.; A Comparative Study of Neural Network Based Feature Extraction Paradigm, Pattern Recognition Letters, vol. 20(1), pp. 7-14, 1999.

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

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

[10] Partridge, D. and Sharkey, N.; Neural computing for software reliability, Expert Systems, 11(3), pp. 167-176, 1994.

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

[12] Partridge, D. and Krzanowski, W.; Distinct Failure diversity in Multiversion Software, Technical Report, 348, Deoartment of Computer Science, Exeter University, 1997.

[13] Bauer, E.; An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning*, 36, 105-142, 1999.

[14] Bigus, J. P.; Data Mining with Neural Networks. New York, Mc-Graw Hill, 1996.

[15] Cohn, P.; Elements of Linear Algebra, Chapman and Hall, 1994.

[16] Cohn, P.; Algebra: Volume 1. , John Wiley and Sons, 1974.