Automatic generation of initial weights and target outputs of multilayer neural networks and its application to pattern classification

メタデータ	言語: eng 出版者: 公開日: 2017-10-03 キーワード (Ja): キーワード (En): 作成者: メールアドレス: 所属:
URL	http://hdl.handle.net/2297/18411

ICONIP'98
The Fifth International Conference on Neural Information Processing
Kitakyushu, Japan October 21-23,1998

Automatic Generation of Initial Weights and Target Outputs of Multilayer Neural Networks and its Application to Pattern Classification

Kanad Keeni †, *Kenji Nakayama* †*and Hiroshi Shimodaira* ††
Email:keeni@ec.t.kanazawa-u.ac.jp

†Dept. of Elec. & Comp. Eng., Kanazawa University
2-40-20 Kodatsuno, Kanazawa 920, Japan
††School of Information Science, JAIST
1-1 Asahidai, Tatsunokuchi, Ishikawa 923-12, Japan

ABSTRACT

This study high lights on the subject of weight initialization in back-propagation feed-forward networks. Training data is analyzed and the notion of *critical point* is introduced for determining the initial weights and the number of hidden units. The proposed method has been applied to two-class classification problem and character recognition problem. The experimental results indicate that due to the constraints imposed during weight initialization and target output representation, the proposed method results in better generalization.
KEYWORDS: Critical points, initial weights, target outputs

1. Introduction

Neural networks architectures have sparked of great interest in recent years because of their intriguing learning capabilities. Several learning algorithms have been developed for training the networks and out of them Back Propagation [1] is probably most widely used. However, the standard back propagation algorithm has the following drawbacks.

1. Learning procedure is time consuming.
2. It is not clear as to how to construct an appropriate feed-forward architecture.

In case of 1., the neural network criterion function is the function of the tunable parameters that the learning algorithm attempts to minimize. Since a Mean Square Error criterion function usually has several local minima, initial values of the parameters influence the final parameters. Several researchers have designed systems in which weights are initialized so that the initial activity of the network corresponds to the successive rules, that may come from an expert. Wilson has proposed Fast BPN [2], where the initial parameters are determined by estimating the signal rank with generalized likelihood ratio test (GLRT) and the singular value decomposition (SVD) of the GLRT covariance matrix. However, the disadvantage of their method is that the number of hidden nodes cannot exceed the input feature dimension. In order to tackle 1, most of the researches have mainly focused on improving the optimization procedure by dynamically adapting the learning rates [3], [4].

In case of 2., the problem has been treated in various ways. One most common approach has been to start with a large number of hidden units and then prune the network once it has trained [5], [6]. However, pruning does not always improve generalization. Another strategy for finding a minimal architecture has been to add or remove units sequentially [7], [8].

In the present study, the above mentioned draw-backs of back propagation has been carefully investigated and a method has been proposed for determining the initial weights for input to hidden layer and the number of hidden units automatically.

2. Initial weights

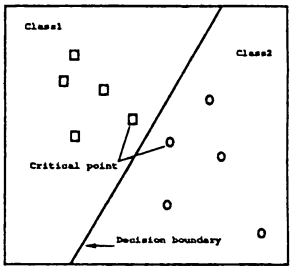


Figure 1: Critical points and decision boundary

Feed forward multi-layer networks that employ back-propagation for training are considered in this study. The decision rule is to select the class corresponding to the output neuron with the largest output. For the sake of simplicity, the number of output unit is set to two (two-class classification problem). However, the concept can be hopefully extended to multi-class classification problems. The decision boundary for a multi-layer feed-forward network is defined as follows.
Definition 1. The decision boundary between two classes in a feed-forward neural networks is the locus of points where both of the output neurons produce the same activation. If we define the activation of output unit i as $O_i(X)$, where X is an input vector and let $d(X) = O_1(X) - O_2(X)$, then the decision boundary can be defined as

$$\{X | d(X) = 0\}$$

Next, the notion of *Critical points* is introduced as follows.
Definition 2. The set of critical points contains pairs of points that satisfy the following conditions:

$$\min_k(d(p_i, q_k)) = d(p_i, q_j), \min_k(d(p_k, q_j)) = d(p_i, q_j)$$

where $d(p_i, q_k)$ denotes the Euclidean distance between the vector p_i and q_k.

If we denote the samples in class ω_1 as p_i and samples in class ω_2 as q_j, then for each sample in class ω_1 and class ω_2,

the set of critical points C can be defined as

$$C = \{(p_i, q_j) | \min_k(d(p_i, q_k)) = d(p_i, q_j),$$

$$\min_k(d(p_k, q_j)) = d(p_i, q_j), p_i \in \omega_1, q_j \in \omega_2\}$$

Now, as shown in Figure 1, the decision boundary must pass through the critical points. If the coordinate of the pair of critical points (p_i, q_k) are (x_i, y_k) and (u_i, v_k) then the ideal hyper plane must go through the point $\frac{(x_i+u_k)}{2}, \frac{(y_i+v_k)}{2}$.

Since, the weight vectors are orthogonal to the separating hyper-plane, the initial weights are generated in the following way. First, the pair of critical points are determined from the training data as mentioned above. Next for all pair of critical points (p_i, q_k), the weight vectors m_n are generated by the following equation:

$$m_n = \frac{p_i - q_k}{\|p_i - q_i\|}$$

and the biases θ_n are generated by the following equation:

$$\theta_n = -n^t \cdot P = -\frac{(p_i - q_k)^t}{\|p_i - q_k\|} \cdot \frac{(p_i + q_k)}{2}$$

3. Database

The proposed method has been applied to the cancer data base obtained from the University of California Machine Learning Repository(two-class classification problem) and Devanagari Characters (multi-class classification problem). The cancer database is publicly available and it contains 699 instances, each having 9 attributes. The Devanagari database consists of 44 characters (23 consonants, 13 c-c combinations, 3 vowels, 2 special characters and 3 numerals). Sixty seven Devanagari characters taken from the text [9] are shown in Figure 3. The characters are stored in the form of 16 × 16 floating point images. A details about the Devanagari database can be found in [10].

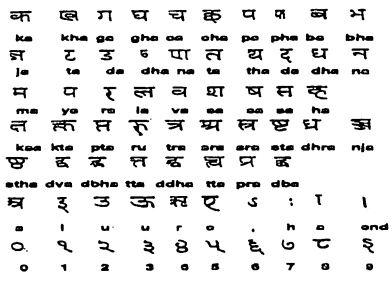


Figure 2: Some of the basic Devanagari characters

4. Experiments

4.1. Two-class classification problem

Here, the proposed weight initialization method is applied to the cancer data base. The whole dataset was divided into ten training and ten testing sets and a ten fold cross validation was performed. In order to evaluate the effectiveness of the proposed method, another set of experiment was performed by applying the standard back-propagation. The number of hidden unit was set to the number of critical points given by the proposed method. The average accuracy rate of the proposed method, standard back-propagation and the results of applying Bayes decision (assuming normal distribution for each category) rule is shown in Table 1.

Method	Accuracy (%)
Proposed	96.8
Standard BP	96.3
Bayes	95.27

Table 1: Average accuracy rate

On the other hand, the following linear programming methods for pattern classification: Multi Surface Method [11] (MSM), Robust Linear Programming [12] (RLP), and Perturbed Robust Linear Programming [13] (RLP-P) were also applied on the same dataset and the results are summarized in Table 2. It can be seen in Table. 3 that out of the three

Method	Accuracy (%)
MSM	92.6
RLP	96.4
RLP-P	96.6

Table 2: Average accuracy rate of linear programming methods

methods RLPP gives the best accuracy of 96.6 %. Now if compared to Table. 2, it is clear that the proposed method gives slightly better result than RLPP.

4.2. Multi-class Classification problem

Here the proposed method is applied to Devanagari characters for evaluating the applicability of the same to character recognition problem. However, a different approach has been taken for representing the output layer.

4.3. Automatic coding scheme

One of the important aspect of information processing is the way the information is represented. The conventional approach of assigning a category to each cell is reasonable in the sense that the distance among the codes is constant. However, this kind of approach can not be taken for representing a large number of categories. Although binary representation appears to be reasonable in the sense that it would require $\log_2 n$ bits for representing n categories; they are not well suited for network output representations because of the interdependency of cells which make the learning difficult.

In the proposed method, at first each training data x_n $(n = 1, \cdots, N)$ (16 × 16 pixels) is split into four parts (8 × 8 pixels) in the following way.

$$x_n = (y_{n1}, y_{n2}, y_{n3}, y_{n4})$$

As a result the following set of pattern is generated for each part $i = 1, \cdots, 4$.

$$Y_i = \{y_{1i}, y_{2i}, \cdots, y_{Ni}\}$$

Next, each Y_i $(i = 1, 2, 3, 4)$ is clustered. Here the LBG method [14] was applied for clustering and the number of cluster was set to 16. After clustering, for each ω_l

$(l = 1, \cdots, L)$ the cluster to which the part belongs is checked and the final cluster is determined by considering the maximum number of characters belonging to a specific cluster. So, in this way, for each category ω_l there will be a four dimensional vector $q_l = (q_{l1}, q_{l2}, q_{l3}, q_{l4})$, where each element of the vector will correspond to the cluster to which the corresponding part of the training sample belongs. Now, each element of vector q_l is mapped to a 5-bit binary pattern. The binary patterns are hand picked and they are mapped in a way such that for any two clusters whose parent is the same, the Euclidean distance between the binary patterns becomes one. This will somehow ensure that the similar part (q_l) of each category ω_l would get similar codes. On the other hand, neglecting the binary patterns containing only 0's or 1's, the minimum number of bits required to maintain this constraint is 5. The code assignment procedure is shown in Figure 3. Finally, q_l is transformed to a feature vector which is a 20-bit

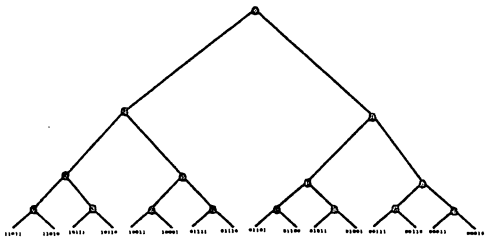


Figure 3: Code assignment procedure

code. In this way for each training sample x_n $(n = 1, \cdots, N)$ there will be a feature vector $c_{(n)}$ with 20 elements. (Here (n) represents the category number to which x_n belongs.) The entire process is summarized in Figure 4.

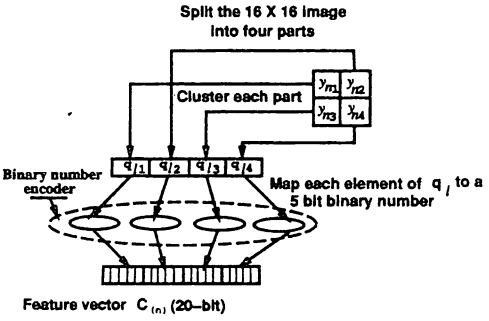


Figure 4: Automatic coding procedure

4.4. Recognition process

The network is totally connected and it consists of one input, one output, and a hidden layer. Here the training process is divided into two phases. In the first phase, the network is trained with the training data in a usual way. In the second phase, the training data is fed to the network and the output vectors given by the network are employed for forming prototypes for each category. The prototype formation process is as follows.

$$m_l = \frac{1}{N_l} \sum_{x \in \omega_l} o$$

Here, N, o, and x stand for the number of samples used for each category during training, the output vector, and the input data respectively. These prototypes are later used for recognition.

During evaluation, Euclidean distance is taken among the prototypes formed in the second phase of training and the output vector given by the network during test. The whole process, from formation of prototype vectors to evaluation is summarized in Figure .

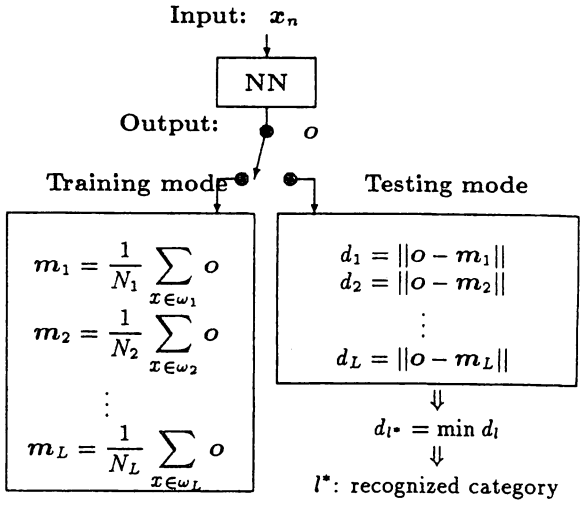


Figure 5: Training and recognition process

Here, the target outputs iare generated by employing the proposed automatic coding scheme. This solves the problem of output layer representation. The next issues are related to network architecture and initial weights.

Instead of performing trial and error, the number of hidden unit is determined by employing the knowledge of critical points. In this case the straight forward approach of setting the number of hidden unit to the number of calculated critical points could not be applied due to the enormous number of critical points.

Therefore, the pair of critical points are sorted in an increasing order based on the distance between each pair. Next, the number of hidden unit is determined by fixing a threshold value d_{max} over the distance. In this way, if d_{max} set to x results in u hidden units, then the first u critical points are employed for calculating the initial weights and biases. This way of selecting the critical points is reasonable in the sense that the patterns that stay close to each other would effect the mean square error of the network during training.

4.5. Results

The relation of d_{max} and the number of hidden unit is shown in Table 3. In order to evaluate the effect of initial weights, one set of experiments were performed by employing only automatic coding procedure (Method1), and another set of experiments were performed by employing both of automatic coding and automatic initial weight generation procedure (Method2).

d_{max}	2.0	2.2	2.4	2.5	3.0
# Hidden Unit	35	46	57	61	76

Table 3: Relation of d_{max} and # hidden unit

However, in both of the cases, the learning parameters were

kept the same. The experimental outcomes are summarized in Figure 6, 7, and 8.

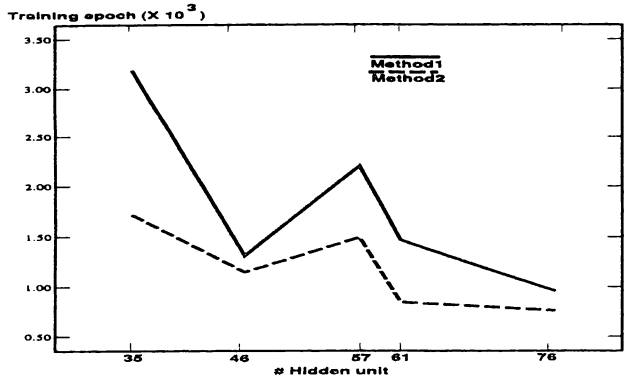


Figure 6: Training epoch (Method1 Vs Method)

It is clear from Figure 6, that the number of training epoch required for the combination of the proposed methods is less than the case where only automatic coding procedure is employed.

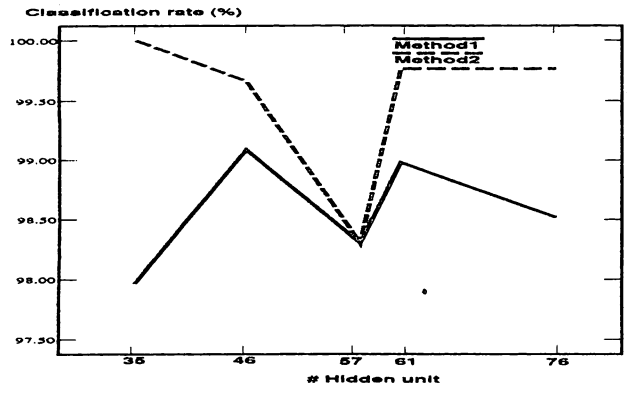


Figure 7: Classification rate (Training data : Method1 Vs Method2)

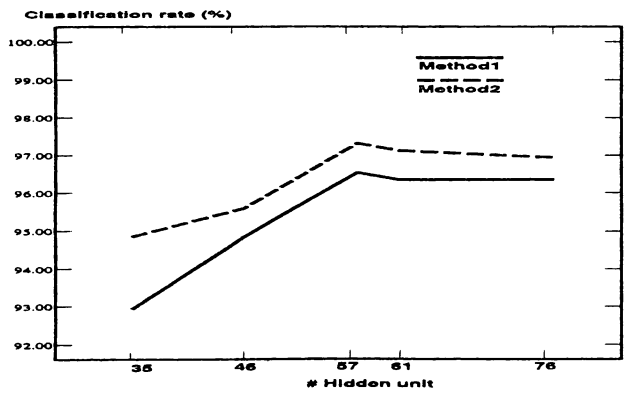


Figure 8: Classification rate (Testing data : Method1 Vs Method2)

In Figure 7 and 8 it can be seen that the combined method results in better performance against both of training and testing data. Now, both of method1 and method2 gave the same classification accuracy for training data with 57 hidden units(Figure 7). However, compared to method1 method2 gave better performance with respect to testing data (Figure 8). This assures that the combined method (method2) produces a better solution. At the same time, the results also indicate that the initial weights are effective for faster training and better solution.

5. Conclusion

A method for generating initial weights and automatic encoding of target outputs have been proposed. Experimental results show that the method results in faster learning and better generalization. The notion of *critical points* has been introduced for determining the number of hidden units.

The method has to be further extended by investigating some other criterion for selecting the pairs of critical points.

References

[1] Rumelhart, McClelland, and the PDP Research Group, "Parallel Distributed Processing," *The MIT Press*, 1986.

[2] David H. Kil, Frances B. Shin, "Pattern recognition and Prediction with applications to Signal Characterization, " *AIP PRESS*, pp. 134-138, 1996.

[3] Riedmiller, Martin and Braun, "RPROP - A fast Adaptive learning algorithm," Technical report *Universitat Karlsruhe*, 1992.

[4] G. Thimm, P. moerland and E. Fiesler, "The interchangeability of learning rate and gain in back propagation neural networks," *Neural computation*, 1996.

[5] M. C. Mozer, P. Smolensky, "Skelitonization : A technique for trimming the fat from a network via relevance assessment," in *Advances in neural information processing systems*, 1, pp. 107-115, 1989.

[6] J. Sietsma and Dow, "Neural network pruning - why and how," Proceeding of the second international conference on neural networks, pp. 326-333, July 1988.

[7] Fahlman, E. Scott, "An empirical study of learning speed in back propagation networks," Technical report *CMU-CS-88-162*, 1988.

[8] T. Ash, "Dynamic node creation in back propagation networks," *Connection Science*, 1(4), pp. 365-375, 1989.

[9] H. Kern and Bunyiu Nanjio, "Saddharmapundarika," *ST.-PETERSBOURG*, 1912.

[10] K. Keeni, H. Shimodaira, T. Nishino, Y. Tan, " Recognition of Devanagari Characters Using Neural Networks," *Transaction of Information and Systems Society of Japan*, IEICE Trans. Inf. &Syst., Vol. E79-D, No. 5, May 1996.

[11] O. L. Mangasarian, "Multi-surface method of pattern separation," *IEEE Transactions on Information Theory*, IT-14, pp. 801-807, 1968.

[12] K. P. Bennett and O. L. Mangasarian, "Robust Linear Programming Discrimination of two Linearly Inseparable sets," *Optimization Methods and Software*, Vol. 1, pp. 23-34, 1992.

[13] O. L. Mangasarian, "Linear and nonlinear separation of patterns by linear programming," *Operations Research*, Vol 13, pp. 444-452, 1965.

[14] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Comm., COM-28*, pp. 84-95, January 1980.