Estimation of Initial Weights and Hidden Units for Fast Learning of Multi-layer Neural Networks for Pattern Classification

Kanad Keeni †, *Kenji Nakayama* ††*and Hiroshi Shimodaira* †††

†Department of Information Systems & Quantitative Sciences, Nanzan University, Japan
††Dept. Elec. & Comp. Eng., Kanazawa University, Japan
†††School of Information Science, JAIST, Japan

ABSTRACT

A method has been proposed for 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 artificial data and the publicly available cancer database. The experimental results of artificial data show that the proposed method takes 1/3 of the training time required for standard back-propagation. In order to verify the effectiveness of the proposed method, standard back-propagation, where the learning starts with random initial weights was also applied to the cancer database. The experimental results indicate that the proposed weight initialization method results in better generalization.

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. The reason for the popularity is the underlying simplicity and relative power of the algorithm. Its power derives from the fact that unlike its precursors, the perception learning rule [2], and the Widrow-Hoff learning rule [3], it can be employed for training nonlinear networks of arbitrary connectivity. Since such networks are often required for real-world applications, such a learning procedure is critical. However, the standard back-propagation algorithm has the following drawbacks.

1. Learning procedure is time consuming - the learning always starts from scratch
2. It is not intuitively obvious as to how to construct an appropriate feed-forward architecture. We are faced with the problem of fixing the number of hidden units for a particular problem.

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. It is widely known that the initial weights largely effect the generalization performance. For example, two networks with same architecture, when trained with totally different initial weights would produce different results. 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 [4], 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 the fact 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 [5] - [6].

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 [7], [8]. However, pruning does not always improve generalization. Another strategy for finding a minimal architecture has been to add or remove units sequentially [9], [10].

On the other hand, it is also well known that neural networks do not make any assumption about probability distribution functions of data and can solve complex problems with arbitrary decision boundary. Therefore, it is desirable to investigate the learning characteristics of the networks for finding an estimate about the decision boundary. It has been shown in [11] that training data selection largely affects the learning process.

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.

This paper is divided into 5 sections. The next section describes the pattern mapping criterion of MLNNs. The third section presents the automatic method for generating initial weights for input to hidden layer. Experimental results of artificial, real world data are provided in the Fourth section. Finally the last section is devoted to conclusion and further researches.

2. Pattern classification characteristics of MLNN

In any pattern classification system, pattern mapping or pattern classification is equivalent to dividing an N dimensional space where the patterns are distributed. In case of multi-layer neural networks (MLNN), this N dimensional space is divided by forming hyper-planes with the help of synaptic weights of nonlinear neurons. MLNNs do not make any assumption about probability distribution functions of data and can solve complex problems with arbitrary decision boundary. The degree of freedom in placing the decision boundary is very high. Therefore, neural networks are considered to be a good choice for pattern classification tasks.

Another most important aspect of neural networks is learn-

ability. In case of supervised learning the networks can find optimal synaptic weights through learning. However, since the neural networks are nonlinear systems and gradient descent is used to find a set of weights to optimize the performance on a particular task, there is always a possibility of getting stuck in local minima. Therefore, global minima or optimal solution is not always guaranteed. Furthermore, the learning process is time consuming and it is highly dependent on the problem that is to be solved.

If we assume that there are no overlaps among the distribution of training patterns then pattern mapping can be categorized in the following classes.

1. $\|x_i - x_j\|$ is small $\bigwedge$ $\|y_i - y_j\|$ is small
2. $\|x_i - x_j\|$ is small $\bigwedge$ $\|y_i - y_j\|$ is large
3. $\|x_i - x_j\|$ is large $\bigwedge$ $\|y_i - y_j\|$ is small
4. $\|x_i - x_j\|$ is large $\bigwedge$ $\|y_i - y_j\|$ is large

Here, x_i is the input vector and y_i is the corresponding output vector, and $\|\cdot\|$ stands for the Euclidean norm. In case of 1., the problem is to map similar input vectors in a way such that the output vectors also become similar. In the second case the input vectors are similar but they are to be mapped as different patterns in the output space. The third case means that the input patterns stay far from each other but they are to be mapped as similar patterns. Finally, the 4th case means that the input patterns are far from each other and they are to be mapped as different patterns. Now, 1., 3., and 4. are not that difficult. However, in case of 2., the problem is to map the patterns that stay very close in the input space, as different patterns in the output space. This means that even though the solution exists, due to the difficulty of the problem the training process would be time consuming.

For example, if we define connection weight from the i'th input to the j'th hidden unit as w_{ij} then the total input and output of the j'th hidden unit can be defined as follows.

$$net_j = \sum_{i=1}^{n} w_{ij} x_i + \theta_j$$

$$O_j = \sigma(net_j)$$

$$\sigma(net_j) = \frac{1}{1 + e^{-net_j}}$$

where, $\sigma(\cdot)$ is the activation function and θ_j is the bias. At the same time the total input to the k'th output unit and the corresponding output can be defined as follows.

$$net_k = \sum_{j=1}^{j} w_{jk} O_i + \theta_k$$

$$O_k = \sigma(net_k)$$

where, $\sigma(\cdot)$ is the same activation function as it was with the hidden layer.

As mentioned earlier the similar patterns play a critical role in learning. Suppose we have training patterns x_{1n} and $x_{2\acute{n}}$ that are very close in the input space and the patterns belong to the class ω_1 and ω_2 respectively. In this case, the network output would become extremely sensitive. This is because the network output must change rapidly for a small change in the input.

Now, if the decision boundary is far from the patterns x_{1n} and $x_{2\acute{n}}$, then the corresponding outputs would have the value $O_{1n} \cong O_{2\acute{n}} \cong 0$ or 1. However, during the learning process, as the decision boundary approaches x_{1n} and $x_{2\acute{n}}$ the output of the corresponding patterns approach the same value, and the learning process becomes extremely slow. In this case, as the decision boundary moves close to the pair x_{1n} , $x_{2\acute{n}}$ or enters the region between the pair, the amount of weight correction becomes extremely small. To be specific, if we assume $O_{1n} \cong O_{2\acute{n}} \cong$ some value y then the amount of correction for the n'th pattern Δ_n would be as follows.

$$\Delta_n = \eta \delta_n O_{nj},$$

$$\delta_n = (t_n - O_n) \acute{f}(net_n),$$

where, O_{nj} is the output of the j'th hidden unit. Now, as the patterns x_{1n} and $x_{2\acute{n}}$ are similar, the output of the jth hidden unit would also become similar, that is

$$O_{1nj} \cong O_{2\acute{n}j},$$

and

$$\acute{f}(net_{1n}) \cong \acute{f}(net_{2\acute{n}}).$$

In this case, the weight correction will be as follows.

$$\Delta_{1n} + \Delta_{2\acute{n}} = \eta O_{1nj}((t_{1n} - O_{1n}) + (t_{2\acute{n}} - O_{2\acute{n}}))\acute{f}(net_{1n}).$$

If it is assumed that the targets of the patterns are

$$t_{1n} = 1, t_{2\acute{n}} = 0,$$

and the output of the patterns are

$$O_{1n} = z, O_{2\acute{n}} = z - \epsilon,$$

then the weight correction would become as follows.

$$\begin{aligned}\Delta_{1n} + \Delta_{2\acute{n}} &= \eta O_{1nj}((t_{1n} - z) + (t_{2\acute{n}} - (z - \epsilon)))\acute{f}(net_{1n}) \\ &= ((t_{1n} + t_{2\acute{n}}) - 2z + \epsilon)\acute{f}(net_{1n}) \\ &= (1 - 2z + \epsilon)\acute{f}(net_{1n})\end{aligned}$$

Now, at the beginning of training, the decision boundary would be far from x_{1n} and $x_{2\acute{n}}$ and in that case the correction of synaptic weights would not be small. However, during the training process, as the decision boundary moves towards x_{1n} and $x_{2\acute{n}}$, because of the similarity of the patterns the output would approach the same value. The most critical situation would take place as z and $\|\epsilon\|$ approach the value 0.5 and 0 respectively. That is,

$$\Delta_{1n} + \Delta_{2\acute{n}} = \lim_{z \to 0.5} \lim_{\epsilon \to 0} (1 - 2z + \epsilon)\acute{f}(net_{1n}) \cong 0$$

Therefore, the correction of weights for these patterns would become very small and as a result the learning process would become extremely slow.

On the other hand, if the patterns x_{1n} and $x_{2\acute{n}}$ are far from each other in the input space, even if the decision boundary moves towards them the activation of the corresponding outputs would not become the same at the same time. Hence, the weight correction will not become small.

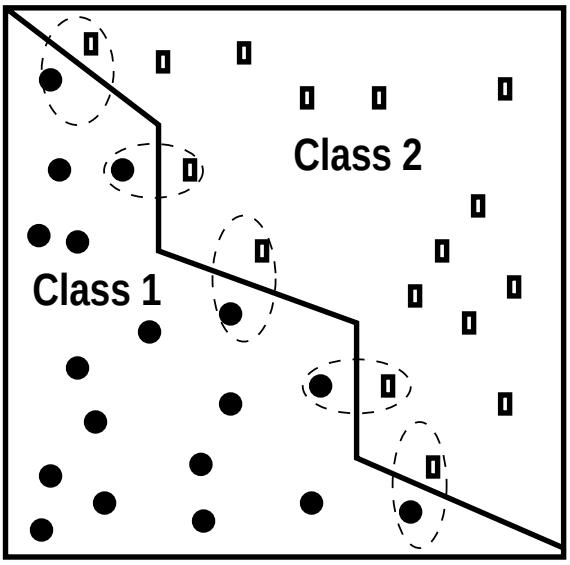

Figure 1: Critical points and decision boundary

3. Estimation of decision boundary

In the present study, feed forward multi-layer networks which use back-propagation for training is considered. 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 output neurons produce same activations

If we define the activation 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 we introduce the idea of *Critical points* 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 Fig. 1, the decision boundary must pass through the critical points. In other words, a hyper-plane has to be placed in between the pair of critical points. If the coordinate of the pair of critical point (p_i, q_k) are (x_i, y_k) and (u_i, v_k) then the ideal hyper plane must go through the point $\frac{(x_i+u_k)}{2}, \frac{(y_i+v_k)}{2}$ and the slope z of the straight line can be calculated from the following equation.

$$\frac{v_k - y_k}{u_i - x_i} \times z = -1$$

In the present approach instead of starting from scratch the initial weights for the hidden units are calculated from the critical points.

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:

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

In the present approach these weights and biases are used as a priori for speeding up learning.

4. Experiments

4.1. Experiments with artificial data

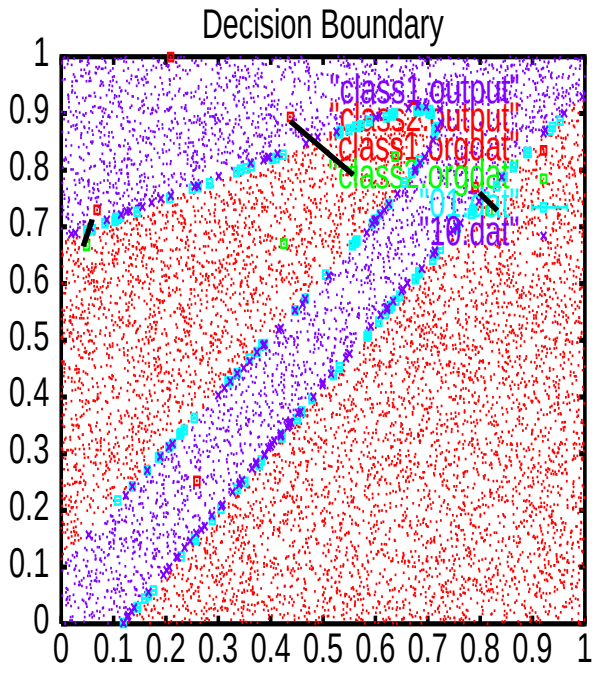

Figure 2: Decision boundary given by proposed method

In the present approach the number of hidden units is kept the same as the number of critical points calculated from the training data. Two dimensional data is used for training and testing. Five samples for each class (in this case 2 classes)

were randomly generated for training. The network had two input units, two output units and the number of hidden unit was set to 3. Training was continued until the mean square error reach 0.001. For testing, 10000 samples were randomly generated and the class to which the testing sample falls is decided by considering the maximum activation of the output units. The network could learn the training data with 3 hidden units. The decision boundary is estimated from output activation of the network in respect to the testing samples as follows.

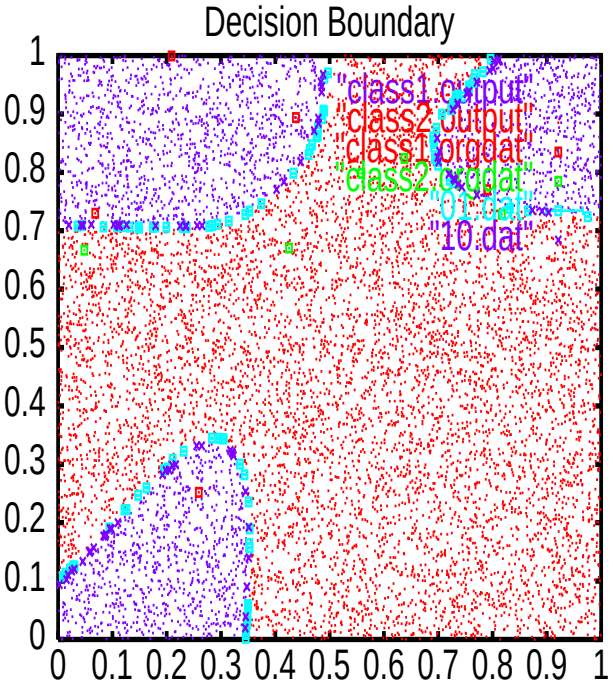

Figure 3: Decision boundary given by Conventional Back-propagation

For each testing sample correctly classified as class ω_1, find the nearest testing sample correctly classified as class ω_2. The same process is repeated for the testing samples classified as class ω_2. Now the line connecting the pairs mentioned above must pass through the decision boundary since the pair of samples correctly classified differently. The decision boundary given by the network is shown in Fig. 2. In Fig. 2, the calculated critical point pairs are connected by a line. In order to evaluate the effectiveness of the proposed method another set of experiments were performed by employing conventional back-propagation algorithm and the decision boundary is shown in Fig. 3. Next, the network was trained with five different initial weights (weights for hidden to output unit connection) , the and the result is summarized in Table. 1.

Init weight	# Epoch Proposed method	# Epoch Standard BP
Seed1	6345	21279
Seed2	6341	21347
Seed3	6251	21263
Seed4	6179	21050
Seed5	6224	22019
Mean	6268	21392

Table 1: Comparison of results

It can be seen in Table. 1., that the iterations necessary for the proposed method is less than 1/3 of that of standard back-propagation. In case of standard back-propagation, there is no other way than to cut and try for determining the number of hidden units necessary for solving a problem. In case of the proposed method, the number of hidden unit is determined automatically.

4.2. Selection criterion of critical points

The straight forward approach of setting the number of hidden units to the number of calculated critical points could not be applied due to the enormous number of critical points. As it has been mentioned previously, the critical points are the points that stay very close to each other and effect the whole learning process in a great extent. Therefore, the first criterion for selecting the pair of critical points based on the minimum distance among all the pairs is reasonable. However, this kind of approach is local, in the sense that a large number of critical points where the distance among each pair is very small may appear very close to each other in the input space. Now, if the characteristic of the hyper-planes formed by the sigmoid function is considered, it is unrealistic to place a hyper-plane for each of these critical points. Therefore, some criterion for rejecting the unnecessary hyper-planes is necessary.

Based on the idea that the input patterns that are very close to each other would slow down the training process, the following approach can be considered for rejecting the unnecessary hyper-planes.

Suppose, we have two hyper planes P_a and P_b as shown in Figure 4. Now, if we express the equation of the hyper-planes as:

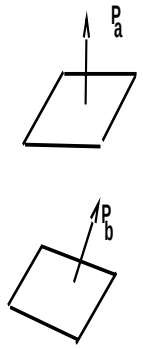

Figure 4: Correlation of hyper-planes

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n + a_0 = 0$$

$$b_1 x_1 + b_2 x_2 + \cdots + b_n x_n + b_0 = 0$$

and the component vector of a of P_a and b of P_b as

$$a = (a_1, a_2, \ldots, a_n) \text{ and}$$

$$b = (b_1, b_2, \ldots, b_n)$$

respectively, where,

$$\sum_{i=1}^{n} a_i^2 = 1$$
$$\sum_{i=1}^{n} b_i^2 = 1.$$

In this case, the first pair of (p_i, q_k) is selected based on the minimum distance among all pairs of critical points. In the next step, the previously selected critical points pair is ignored and the correlation of the previously selected pair and all the other remaining critical points are considered in the following way.

Suppose, P_a is the hyper plane calculated from the first pair of critical points (p_i, q_k) and Q_b is the hyper-plane with which the correlation of the first hyper-plane is to be compared. So, in this case we will have two hyper-planes as shown in Figure 4.

In this case, the correlation of P_a and Q_b can be defined as:

$$D(Pa, Q_b) = \frac{a^t b}{\|a\|\|b\|}$$

In the present approach, if $D(P_a, Q_b)$ is $< max_\theta$ then Q_b is merged with P_b otherwise Q_b is also selected and the process is repeated for all the other remaining critical points.

4.3. Experiments with Real Data

Init weight	# Epoch Proposed method	# Epoch Standard BP
Seed1	131	1771
Seed2	131	2566
Seed3	128	2551
Seed4	130	1270
Seed5	129	3215
Mean	130	2275

Table 2: Comparison of training epoch

Experiments were performed by using the cancer data base obtained from the University of California machine learning database. The database is publicly available and it contains 699 instances, each having 9 attributes. It was divided into training (60%) and testing sets (40%) and five experiments were performed by setting the value of the seed to five different values. Here, the value of $< max_\theta$ was set to 26 degree. This angle is determined by experience. The performance of the proposed method has been compared with the standard back propagation where the learning starts with small randomly distributed initial weights. The experimental outcomes are are summarized in Table 2 and Table 3. It can be seen in Table 2 that the proposed method takes less training epochs compared to standard BP. On the other hand it is shown in Table. 3 that the proposed method results in slightly better performance.

Classification rate	Proposed method	Standard BP
Training data	97.71	97.71
Testing data	96.42	96.06

Table 3: Average accuracy rate (%)

5. Conclusion

It has been successfully shown through experiments that the a priori related to decision boundary can be employed for determining the initial weights of the network. Compared to standard back-propagation the proposed method reduces training time. The method has been successfully applied to the publicly available cancer database. On the other hand the method determines the number of hidden units automatically.

However, optimality of the number of hidden units determined by the proposed method is yet to be investigated. The method has to be applied to other pattern classification problems.

References

[1] Rumelhart, McClelland, and the PDP Research Group, "Parallel Distributed Processing," *The MIT Press*, 1989.

[2] Rosenblatt,F : Two Theorems of Statistical Separability in the Perceptron;*Proceedings of a Symposium on the Mechanization of Thought Process, Her Majesty's Stationary Office, London*,1959,421-456.

[3] Widrow, B., and Hoff, M.E: Adaptive Switching Circuits;Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record,part4,1960,96-104.

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

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

[6] Y. Riedmiller, Martin and Braun : RPROP - A fast Adaptive learning algorithm; Technical report *Universitat Karlsruhe*, 1992.

[7] 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.

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

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

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

[11] K. Hara and K. Nakayama: Training Data Selection Method for generalization by multi-layer Neural Networks;*Transaction of Information and Systems Society of Japan, Fundamentals*, VOL.E81-A, No.3,pp. 374-381, March 1998