Chapter IV
Experiments and Results

4.1 Database and Neural Network Setup

To illustrate how the novel technique works, we will apply it to three examples. The first and the second example concern with the data set in 2-dimensions where the data of the first example are in real space and of the second example are in binary space. The network has only one layer and has only one neuron to classify a given data set. These two examples were appeared in Lursinsap and Tanprasert [11] but with a new method of new weight vector finding gives better results. The third example concerns with the input data set of the neurons of hidden layer are in 2-dimensional binary space and input data set of the neuron of output layer are in real spaces. The network has 2 layers, which are 2 neurons in the hidden layer and 1 neuron in the output layer, classify the given input data. The initial weights are randomized in the range of [-1,1].

4.1.1 Zigzag Problem and AND Problem

The data and boundary vector pairs were presented in [11], when we solved the problem by the method in [11] the new target errors were disturbed. Table 4.1 and 4.2 show the Zigzag problem, the target error before optimizing the weights location is 0.005588 and the target error after optimizing is 0.006875. Table 4.3 and 4.4 show the AND problem, the target error before optimizing the weights location is 0.000998 and the target error after optimizing is 0.01014. This shows that the method in [11] does not ensure the target error after optimizing the weight vector, and our new method will solve this problem.

The selected total sum square error for the target error is 0.006 for Zigzag problem, and 0.001 for AND problem. For each value of the total sum square error, we continue to relocate weight vector with the standard deviation 0.001. The first objective is to achieve the specified total sum square error for the target error. The second objective is to make the total sum square error for the location error less than the location error as the consequence of the first objective while attempting to maintain the target error. The **New Method** in the tables refers to our technique while the **Old Method** refers to the Lursinsap and Tanprasert's technique [11].

The results in Table 4.1-4.4 show that new the technique can enhance the immunity of neuron, $T_{i,j}$ decreases to zero, and it does not disturb the data-fitting errors.

4.1.2 XOR Problem

The XOR problem is one of the benchmark for training algorithm performance comparison and this architecture is a masterpiece for multilayer the perceptron type. The data set consists of four input vectors with their targets which are (0,0,0), (1,0,1), (1,1,0), (0,1,1). The first two elements of a 3-tuples are the input elements while the last element is the target. We assign class A to the data vector having target 1 and class B to the vectors having target 0.

For the hidden layer, there are two decision lines that are l_1 and l_2. Consider line l_1, the boundary vectors in class A are (0,1) and (1,0), and the boundary vector in class B is (1,1). These vectors are, then, used to find the boundary vector pairs for relocating the weight w_1, w_2, and w_5. In class A, vector (0,1) and (1,0) form boundary vector pairs with boundary vector in class B which is (1,1). In class B, vector (1,1) forms boundary vector pairs with vectors (1,0) and (0,1) in class A. Therefore, we only consider these boundary vector pairs, which are (1,1), (1,0) and (1,1), (0,1), to find w_1', w_2', w_5'.

Consider line l_2, the boundary vectors in class A are (0,1) and (1,0), and the boundary vector in class B is (0,0). These vectors are then used to find the boundary vector pairs for relocating the weight w_3, w_4, and w_6. In class A, vectors (0,1) and (1,0) form boundary vector pairs with boundary vector (0,0) in class B. In class B, vector (0,0) forms boundary vector pairs with vectors (0,1) and (1,0) in class A. Therefore, we only consider these boundary vector pairs, which are (0,1), (0,0) and (1,0), (0,0) to find w'_3, w'_4, w'_6.

For the output layer, there is only one decision line, l_3. The inputs are not 0 or 1 but are 0^+ (approaches to 0 in positive direction) or 1^- (approaches to 1 in negative direction). Consider line l_3, the boundary vectors in class A are $(1^-, 1^-)$ and $(0^+, 0^+)$, and the boundary vectors in class B is $(1^-, 0^+)$. These vectors are then used to find the boundary vector pairs for relocating the weight w_7, w_8, w_9. In class A, the vectors $(1^-, 1^-)$ and $(0^+, 0^+)$ form boundary vector pairs with boundary vector $(1^-, 0^+)$ in class B. In class B, vector $(1^-, 0^+)$ forms boundary vector pairs with vectors $(1^-, 1^-)$ and $(0^+, 0^+)$ in class A. Therefore, we only consider these boundary vectors pairs $(0^-, 0^+)$, $(1^-, 0^+)$ and $(1^-, 1^-)$, $(1^-, 0^+)$ to find w'_7, w'_8, and w'_9. The total location error of the network is defined as the summation of the location error of each line which is

$$E_l = E_{l1} + E_{l2} + E_{l3}.$$

We select maximum target error square three values, that are 0.01, 0.001, and 0.0001. The first objective is the target error square that is at least less than or equal to the setting target error, E_{t0}. The second object is the location error square of new weight vector that must be less than the previous generation and it must be less than some positive threshold, E_{lmax}, value in the final. Table 4.5 and Table 4.6 show the results when the maximum target error is set to 0.001. The average convergence rates, from 5 runs, in case of target error 0.01 ,0.001, and 0.0001 with various standard deviation intervals of the random optimization are concluded in Figure 4.1, Figure 4.2, and Figure 4.3 respectively.

Table 4.1: Comparison of E_t and E_l of Zigzag problem obtained from new and old methods.

Error	Before Relocating Weights	New Method	Old Method
E_t	0.005588	0.005580	0.006875
E_l	0.452997	$< 10^{-6}$	0.000006

Table 4.2: Comparison of weight tolerances of Zigzag problem obtained from new and old methods.

Before Relocating		New Method		Old Method	
Weight value	$T_{i,j}$	Weight value	$T_{i,j}$	Weight value	$T_{i,j}$
205.086792	0.185082	194.966812	0.000003	187.874207	0.000878
270.159271	0.246776	292.450745	0.000004	281.664001	0.001229
-30.606279	0.164517	-31.194712	0.000002	-30.052000	0.000878

Table 4.3: Comparison of E_t and E_l of AND problem obtained from new and old methods.

Error	Before Relocating Weights	New Method	Old Method
E_t	0.001	0.000998	0.01014
E_l	0.00956	$< 10^{-7}$	10^{-7}

Table 4.4: Comparison of weight tolerances of AND problem obtained from new and old methods.

Before Relocating		New Method		Old Method	
Weight value	$T_{i,j}$	Weight value	$T_{i,j}$	Weight value	$T_{i,j}$
7.215832	0.049716	7.263439	0.000001	7.247273	0.000066
7.216759	0.494530	7.263455	0.000006	7.246964	0.000019
-10.911733	0.049716	-10.895164	0.000006	-10.870636	0.000066

Table 4.5: Comparison of E_t and E_l of XOR problem obtained from new and old methods.

Error	Before Relocating	Std. = 0.01	Std. = 0.001	Std. = 0.0001
E_t	0.000991	0.00097031	0.00098278	0.00099000
E_l	0.617436	0.00001276	0.00000632	0.00000010

Table 4.6: Comparison of weight tolerances of XOR problem obtained from new method.

w_i	Before Relocating		Std. = 0.01		Std. = 0.001		Std. = 0.0001	
	Weight	$T_{i,j}$	Weight	$T_{i,j}$	Weight	$T_{i,j}$	Weight	$T_{i,j}$
w_1	-6.190063	0.035470	-6.168860	0.000912	-6.107551	0.000169	-6.121018	0.000037
w_2	5.954652	0.000000	6.179140	0.000000	6.107980	0.000000	6.120777	0.000000
w_3	-5.440071	0.00000	-5.343737	0.000000	-5.354218	0.000000	-5.334311	0.000000
w_4	5.178964	0.248308	5.344206	0.000632	5.354396	0.000017	5.334133	0.000015
w_5	3.266494	0.104983	3.084311	0.000912	3.053819	0.000169	3.060573	0.000042
w_6	-2.759335	0.248308	-2.671492	0.000632	-2.677264	0.000049	-2.666998	0.000051
w_7	-8.867808	0.036262	-8.999087	0.000090	-8.964649	0.000039	-8.937285	0.000020
w_8	9.174526	0.818557	9.123857	0.004178	9.157241	0.002681	9.181020	0.002150
w_9	4.190588	0.024938	4.239779	0.000149	4.183619	0.000036	4.146282	0.000019

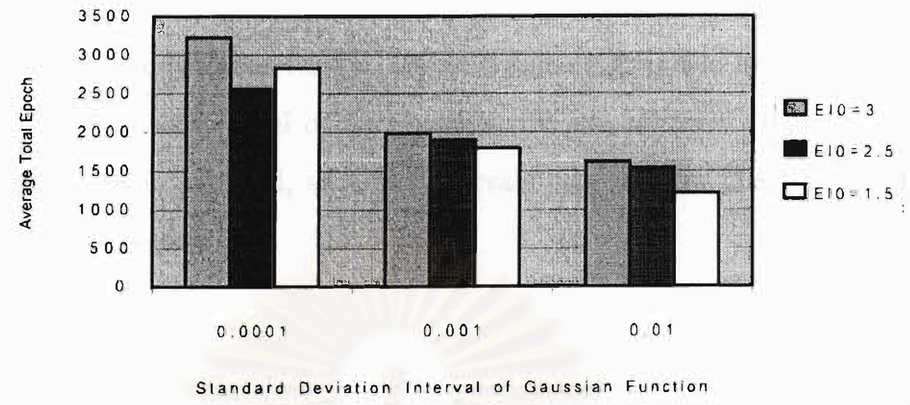

Figure 4.1: Convergent rate at E_t=0.01 with varying σ and location error

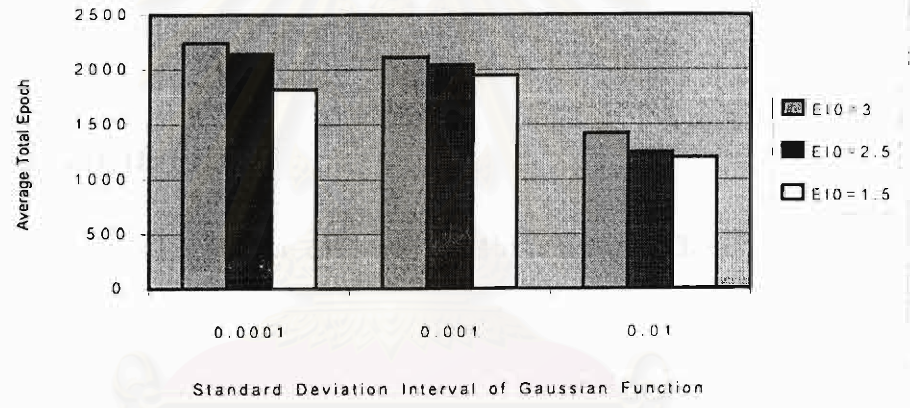

Figure 4.2: Convergent rate at E_t=0.001 with varying σ and location error

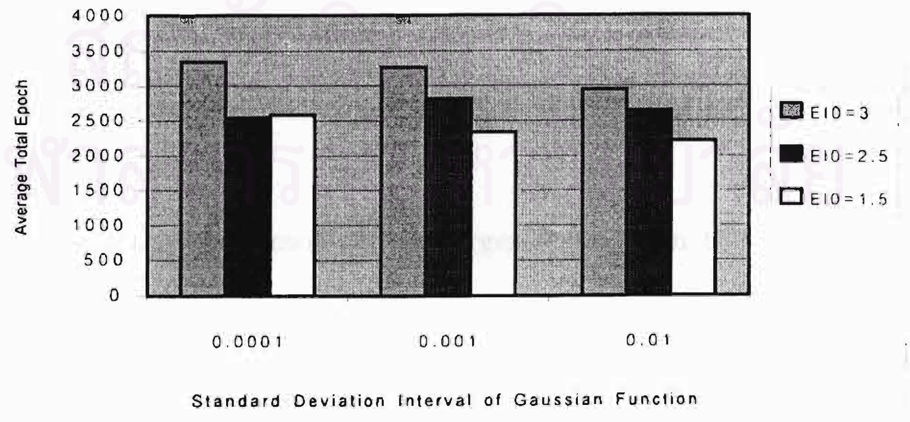

Figure 4.3: Convergent rate at E_t=0.0001 with varying σ and location error

The results from Table 4.5 and Table 4.6 show that the new technique can enhance the immunity level of each weight. When we vary the magnitude of standard deviation interval, smaller intervals can achieve the finer immunity than the larger intervals.

Figure 4.1-4.3 show the dependence of convergent speed and standard deviation interval (σ) when we start with different location errors, E_{l0}. Results of 5 runs show that the higher level of location errors spends more time for convergence than the lower location errors by considering the same magnitude of standard deviation of the random algorithm.

4.2 Discussion

Several related issues will be discussed in this section. These issues include (a) the characteristics of the solution of the objective that results from random optimization algorithm, (b) the technique of how to further increase the tolerance interval based on the hardware implementation, and (c) set of the boundary vectors from the algorithm.

4.2.1 The Characteristics of the Solution

There are several remarkable characteristics of the solution. We will discuss each one as follows:

1. A System that has lower E_l converges faster than the system that has higher E_l.

   It is noticed that the speed of convergence does not depend on E_t but depends on E_l. If $(f_A + f_B)^2$ is not equal to zero, it means $f_A > |f_B|$ or $f_A < |f_B|$, to ensure that E_t does not decrease the new weights must satisfy $f'_A \geq \max(f_A, |f_B|)$ and $f'_B \leq -\max(f_A, |f_B|)$. In addition, to ensure E_l decreases, the new weights must satisfy $(f'_A + f'_B)^2 < (f_A + f_B)^2$. $E'_l(W') < E_l(W)$ means $|f'_A - |f'_B|| < |f_A - |f_B||$. Hence, if $d(W', W^*)$ is less than $d(W, W^*)$, where W^* is the optimal weight vector, the speed of convergence is a proportional of the location error. Consider the step size of the jumping or the interval of the standard deviation of the random algorithm, the algorithm with a larger step will converge faster than the algorithm with a smaller step.

2. Eventually, E_t slightly decreases whereas E_l decreases to zero.

3. *Headroom* as defined in [1] is another tolerance measure for each $w_{i,j}$. If the weight vector relocation minimizes the error $E_t(W) + E_l(W)$, then the value of the *headroom* of each $w_{i,j}$ should be maximized. The value of *headroom* is defined in terms of the percentage change of each perturbed weight and threshold so as to retain the correct classification of the network.

   Let HR be a *headroom* defined as follows:
$$HR = \min\left(\frac{w_{i,j} - l_{i,j}}{|w_{i,j}|}, \frac{u_{i,j} - w_{i,j}}{|w_{i,j}|}\right).$$
   There are two models of fault that we consider. The first model is the link perturbation fault model. Each W_k is perturbed by adding or subtracting the perturbation vector $\delta_{\mathbf{k}}$, i.e. $W_k \pm \delta_{\mathbf{k}}$. The second model is the input vector perturbation. Each input vector X^μ regardless of class is perturbed by adding or subtracting the perturbation vector δ_μ, i.e. $X^\mu \pm \delta_\mu$. We will consider the effect of fault in a two-layer network. We start with the convergent network. The link perturbation fault on output layer can cause only the misclassification. However, if this link perturbation fault occurs on the hidden layer, it will cause both input perturbation fault to the output layer and the misclassification. Table 4.7 shows the headroom of the AND problem. The headrooms of the other example will be shown

Table 4.7: Headroom of AND problem.

Weight	Original Headroom	New Headroom
w_1	0.48790	0.50000
w_2	0.48787	0.50000
w_3	0.32267	0.33322

in [28]. The following theorem addresses the relation among HR, f_A and f_B.

Theorem 5 *For a neuron with binary input vector, if* $f_A \geq 0, f_B < 0$ *and* $f_A + f_B = 0$*, then* HR *is maximized.*

Proof Suppose HR is not maximized, i.e., there exists $1 \leq j \leq n$ such that $\frac{\delta_{A,j}}{|w_{i,j}|} \neq -\frac{\delta_{B,j}}{|w_{i,j}|}$. By lemma 1 we can conclude that $f_A + f_B \neq 0$. By contrapositive, if $f_A + f_B = 0$ then HR is maximized. □

4. The distortion of the input vector is maximized. In this case we will consider when each input vector is slightly different from the training set. If $f_A + f_B = 0$, then it is obvious that the distance of every reference boundary vector pair must be maximized. This implies that the deviation distance of hyperplane is symmetry and the input perturbation constants $\delta x_{A,i}$ of $x_{A,j}$ and $\delta x_{B,i}$ of $x_{B,j}$ are also maximized. The similar concept regarding the equivalent distortion of input vector and weight perturbation was proposed in [26].

4.2.2 Tolerance Interval

The tolerance capability obtained from the previously discussed technique can be further enhanced by carefully considering the tolerance measure of each w_{ij}.

Before having a further discussion, some terms are defined as follows:

Definition 3 *An upper tolerance interval, $U_{i,j}$, of $w_{i,j}$ is $|u_{i,j} - w_{i,j}|$.*

Definition 4 *An lower tolerance interval, $L_{i,j}$ of $w_{i,j}$ is $|w_{i,j} - l_{i,j}|$.*

The tolerance measure discussed in the previous section indicates how each $w_{i,j}$ can symmetrically deviate either increasingly or decreasingly from this current location. Although this measure does not provide us any information regarding the deviating distance from the current location, it gives us freedom to scale up the tolerance intervals $U_{i,j}$ and $L_{i,j}$ to a certain limit as long as the tolerance ratio is constant. This means that the value of each $w_{i,j}$ is also increased by the same scaling factor without affecting the classification capability of the neuron i. We will prove it in the following theorem. Let f_i be the dot product of weight vector W_i with input vector X and $\tilde{f}_i$ equal to σf_i, where $\sigma > 0$ is a scaling factor.

Theorem 6 *Both f_i and $\tilde{f}_i$ correctly classify the input data into classes A and B.*

Proof The classification is defined by the following rules. The data are in class A if $f_i \geq 0$ and in class B if $f_i < 0$. Multiply both sides of the inequality by σ, we still obtain $\tilde{f}_i \geq 0$ for data in class A and $\tilde{f}_i < 0$ for data in class B. □

Generally, a neuron is always implemented by a digital circuit. In this case, the limit of tolerance interval is constrained by the size of register used to store the value of $w_{i,j}$. We assume that, for any neuron, every $w_{i,j}$ uses register of the same size. Let r be the size of a register. Hence, the maximum value to be stored in this register is $2^r - 1$. There are two possible cases. The first case is when $2^r - 1$ is less than $\max_j(u_{i,j})$ and the second is when $\max_j(u_{i,j})$ is larger than $2^r - 1$. $u_{i,j}$ is the upper bound of $w_{i,j}$.

Table 4.8: Lower bound tolerance intervals in each case.

Cases	*Scaling Factor*	*Original $L_{i,j}$*	*New $L_{i,j}$*
$2^r - 1 < \max_j(u_{i,j})$	$\frac{\max_j(u_{i,j})}{2^r-1}$	$w_{i,j} - l_{i,j}$	$\frac{\max_j(u_{i,j})}{2^r-1}(w_{i,j} - l_{i,j})$
$\max_j(u_{i,j}) < 2^r - 1$	$\frac{2^r-1}{\max_j(u_{i,j})}$	$w_{i,j} - l_{i,j}$	$\frac{2^r-1}{\max_j(u_{i,j})}(w_{i,j} - l_{i,j})$

Table 4.9 Upper bound tolerance intervals in each case.

Cases	*Scaling Factor*	*Original $U_{i,j}$*	*New $U_{i,j}$*
$2^r - 1 < \max_j(u_{i,j})$	$\frac{\max_j(u_{i,j})}{2^r-1}$	$u_{i,j} - w_{i,j}$	$\frac{\max_j(u_{i,j})}{2^r-1}(u_{i,j} - w_{i,j})$
$\max_j(u_{i,j}) < 2^r - 1$	$\frac{2^r-1}{\max_j(u_{i,j})}$	$u_{i,j} - w_{i,j}$	$\frac{2^r-1}{\max_j(u_{i,j})}(u_{i,j} - w_{i,j})$

In the first case, we scale the value of $\max_j(u_{i,j})$ down until it is equal to $2^r - 1$. Thus, the scaling factor in this case is equal to $\frac{\max_j(u_{i,j})}{2^r-1}$. Similarly, we scale $\max_j(u_{i,j})$ up until it is equal to $2^r - 1$ for the second case. Hence, the scaling factor in this case is $\frac{2^r-1}{\max_j(u_{i,j})}$. The scaling factor is then used to scale the value of $w_{i,j}$. Tables 4.8 and 4.9 summarize the tolerance intervals in both cases after scaling.

4.2.3 Correctness of Algorithm 1

Algorithm 1 determines the boundary vectors by separately considering each individual $w_{i,j}$. The problem that we were interested in is whether these boundary vectors are different from the boundary vectors obtained by either simultaneously increasing or decreasing all $w_{i,j}$'s. The answer is no. The difference between considering each individual $w_{i,j}$ and simultaneously considering all $w_{i,j}$'s is the first one is based on the assumption of a single fault while the second one is based on multiple faults. The following theorem verifies the answer.

Theorem 7 *The boundary vector sets in cases of single and multiple faults are the same.*

Proof We prove only the case of increasing $w_{i,j}$. The case of decreasing $w_{i,j}$ has the same argument. Without loss of generality, suppose that $w_{i,j}$ is the element that we consider and the input vector is $(x_1, x_2, \cdots x_n)$. Let δ be a small deviation constant for $w_{i,j}$ in a single fault case and δ_k a small deviation constant for $w_{i,k}$ in a multiple fault case. If the boundary vectors in the case of single fault are different from those in the case of multiple fault, the dot product of the single faulty weight vector with the input vector is larger than the dot product of the multiple faulty weight vector with the input vector. Therefore, we have

$$\begin{aligned} w_{i,1}x_1 + \cdots + (w_{i,j} + \delta)x_j + \cdots + w_{i,n}x_n &\geq (w_{i,1} + \delta_1)x_1 + \cdots + \\ &\quad (w_{i,n} + \delta_n)x_n, \qquad (4.1) \\ \delta x_j &\geq \delta_1 x_1 + \delta_2 x_2 + \cdots + \delta_n x_n (4.2) \end{aligned}$$

Under the same environment, δ and δ_k must be monotonically either increased or decreased. Since the smallest values of δ and δ_k are the same, for every k, we divide both sides by δ. Thus the inequality becomes

$$x_j \geq x_1 + x_2 + \cdots + x_n.$$

It can be seen that this inequality is not correct because each x_j is positive. This means that when gradually increasing or decreasing δ_j of each $w_{i,j}$ in case of multiple fault, the weight vector will eventually touch the same boundary vectors. □