Journal of Statistical
Theory & Methods
1997, Vol. 8, No. 2, pp. 111 ~ 118

Improving Noise Tolerance in Hopfield Networks [1]

Youngtae Kim [2] and Jeong Hyun Park [3]

Abstract

Adding a noise tolerance factor to the Relaxation learning algorithm in Hopfield network improves noise tolerance without effecting storage capacity. The new algorithm is called the Pseudo-Relaxation algorithm, and the convergence of the algorithm has been proved. It is also shown that the noise tolerance factor does not effect learning speed.

Key Words and Phrases: Hopfield network, Learning algorithm, Noise tolerance.

1. Introduction

The Relaxation algorithm is an efficient learning algorithm which is derived geometrically from the Fixed increment algorithm by solving the linear inequality system [1]. The Pseudo-Relaxation algorithm is an learning algorithm in which the noise tolerance factor is introduced to the Relaxation algorithm. It is shown that both algorithms have very similar learning speed in terms of weight changes. Depending on the optimal values of the noise tolerance factor, however, the noise tolerance is improved in the Pseudo-Relaxation algorithm. We applied these algorithms for learning and pattern recognition of three different data set (10 digits, 26 upper-case alphabets and 26 lower-case alphabets) of IBM PC CGA character font in a symmetric Hopfield network. The learning speed of two algorithms is compared and the changes of noise tolerance using different values of the noise tolerance factor are described. The role of the noise tolerance factor in the Pseudo-Relaxation algorithm is also analyzed in the geometrical aspect.

[1]본 논문은 1996년 관동대학교 안식년 기간에 연구되었음.
[2]Research Associate Department of Computer Science, Iowa State University, USA
[3]Associate Professor, Department of Computer Science and Statistics, Kwandong University

2. Machine Learning: Relaxation Algorithm

The Relaxation algorithm is also called the Fractional correction algorithm. The perceptron is trained by moving some weight vector $\mathbf{W}$ to the desired side of each pattern hyperplane. Since the most direct path across the hyperplane $\mathbf{H}_j(\mathbf{W}\cdot\mathbf{X}^j = 0)$ is along the normal to it, $|\ \mathbf{W}_i^{k+1}\cdot\mathbf{X}^j - \mathbf{W}_i^k\cdot\mathbf{X}^j\ |$ is proportional to the Euclidian distance between $\mathbf{W}$ and the hyperplane $\mathbf{H}_j$. The learning rate λ is provided as a parameter. The weights are changed in the following manner:

$$\mathbf{W}_i^{k+1} = \mathbf{W}_i^k + \lambda(Y_i^j - O_i^j)\left|\frac{\mathbf{W}_i^k\cdot\mathbf{X}^j}{\mathbf{X}^j\cdot\mathbf{X}^j}\right|\mathbf{X}^j, \quad \text{if } Y_i^j \neq O_i^j$$

where $\mathbf{W}_i^k$	:	Weight array of perceptron i after k changes.
$\mathbf{X}^j$	:	The jth input vector.
Y_i^j	:	Desired output of perceptron i for input $\mathbf{X}^j$.
O_i^j	:	Actual output of perceptron i for input $\mathbf{X}^j$.
λ	:	A parameter in the range $(0, 2)$ which determines the degree of correction.

The theoretical convergence proof is given in [2]. Figure 1 illustrates the convergence of the algorithm geometrically. The shaded area shows the solution set of weights, and each hyperplanes are from the input patterns. The initial weights are randomly generated, and the following weights are decided by moving the weights along the normal to the hyperplanes.

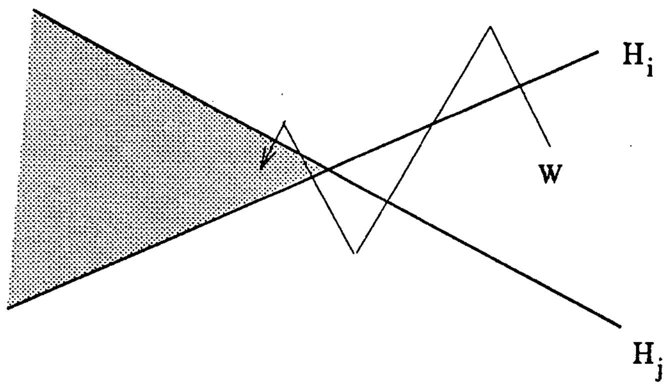

Figure 1: Changes of weight in Relaxation algorithm

3. Noise Tolerance Factor ξ

The noise tolerance factor ξ is introduced to improve the noise tolerance in the Relaxation algorithm. The Relaxation algorithm with the noise tolerance factor ξ is also called the Pseudo-Relaxation algofithm and proposed in reference 3. The algorithm is gives by :

$$\mathbf{W}^{k+1} = \mathbf{W}_i^k + \lambda(Y_i^j - O_i^j)\left|\frac{\mathbf{W}_i^k \cdot \mathbf{X}^j}{\mathbf{X}^j \cdot \mathbf{X}^j - \xi \mathbf{X}_i^j}\right|\mathbf{X}^j, \quad \text{if } Y_i^j \neq O_i^j$$

where $\mathbf{W}_i^k$	:	Weight array of perceptron i after k changes.
$\mathbf{X}^j$	:	The j^{th} input vector.
Y_i^j	:	Desired output of perceptron i for input $\mathbf{X}^j$.
O_i^j	:	Actual output of perceptron i for input $\mathbf{X}^j$.
λ	:	A parameter in the range $(0, 2)$ which determines the degree of correction.
ξ	:	A noise tolerance factor.

Unlikely the Relaxation algorithm, the hyperplane $\mathbf{H}'_j(\mathbf{W} \cdot \mathbf{X}'^j = 0)$ is introduced so that $d(\mathbf{H}'_j, \mathbf{H}_j) = \xi$, where $d(\mathbf{H}'_j, \mathbb{H}_j)$ is the Euclidian distance between $\mathbf{H}'_j$ and $\mathbf{H}_j$. The weight changes in the Pseudo-Relaxation algorithm is illustrated in Figure 2. The solution set S_ξ is depending on the value of ξ. Note that $S_{\xi_i} > S_{\xi_j}$ if $\xi_i < \xi_j$.

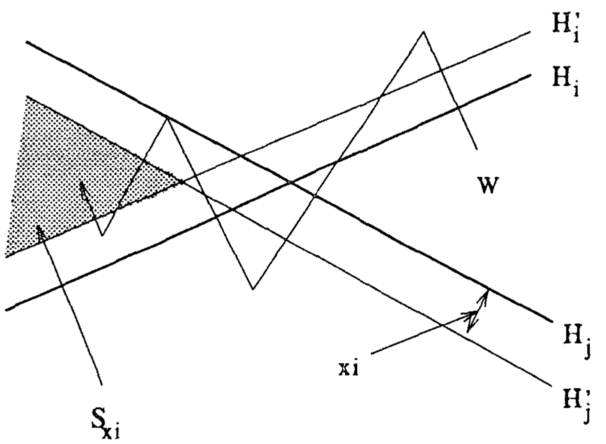

Figure 2: Changes of weight in Pseudo-Relaxation algorithm

A larger value of ξ increases the like-hood of finding a weight point which is deeper in the convex set S_{ξ_i} defined by the traning pattern hyperplanes. In other words, the distance from the weight point to the nearest training pattern hyperplane becomes larger. Adding noise to a training pattern has the effect of perturbing the corresponding hyperplane in weight space. A pattern will be correctly recalled if the weight point of the network still lies on the positive side of this perturbed hyperplane. the further weighet point is from the orginal hyperplane, the more the hyperplanes may be perturbed with the weight point remaining in the positive side. So a larger value of ξ should give the increased noise tolerance over a smaller value of ξ. Figure 3 illustrates the relationship between ξ and the perturbed input pattern.

4. Effect of ξ on Weight Changes and Noise Tolerance

This experimental study was carried out with three different training samples. The three training samples are as follows:

- 10 digits on 7×7 grid (49 units)
- 26 upper-case letters on 7×7 grid (49 units)
- 26 lower-case letters on 7×7 grid (49 units)

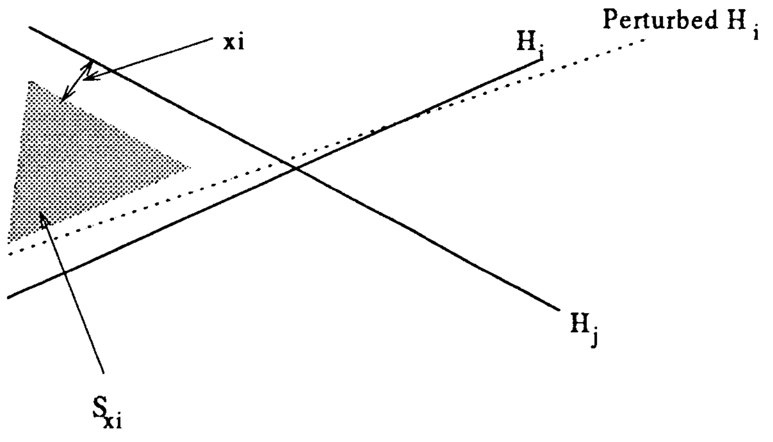

Figure 3: Perturbed pattern and ξ in Pseudo-Relaxation algorithm

The training sets were chosen from IBM PC CGA character font. The network is fully connected, and the units in the networks are bipolar with states 1 and -1. We chose 1.8 for the value of the learning rate λ for the learning algorithms since the optimal learning speed was obtained experimentally using $\lambda = 1.8$.

4.1 Weight changes with different ξ values

It is easily verified that the noise tolerance factor ξ does not effect the weight changes much. Figure 2 describes the effect of ξ on weight changes. A larger value of ξ increases the like-hood of finding a deeper weight point in the solution convex set S_ξ. Since the solution convex set is open, the number of weight changes is not effected by the value of ξ.

Figure 4 shows the weight changes using the ξ value 0.0, 0.1, 1.0, 10.0, 100.0 and 1000.0. We can see that the number of weight changes is not changed much by varying the ξ value. In general, the larger size of the training samples, the more the weight changes required as shown in Figure 4. Specially with the 10 digit training sample, the number of changes is very stable for different ξ's.

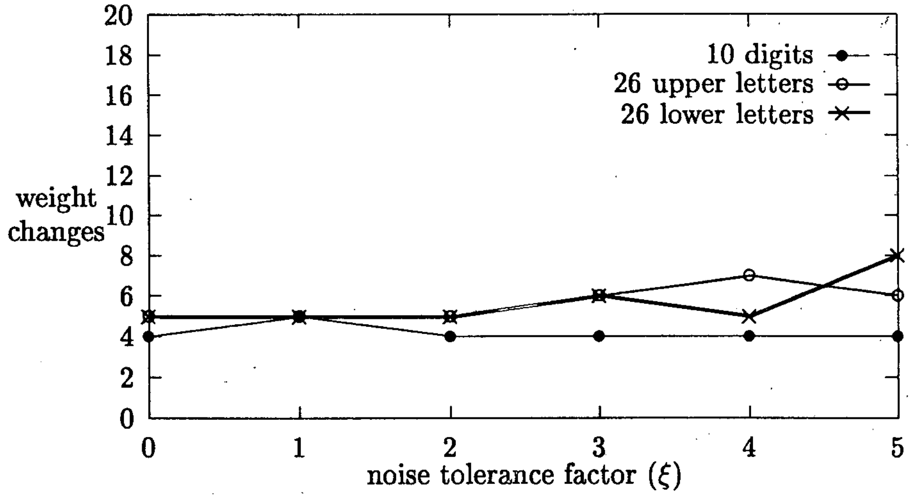

Figure 4: Weight changes using different ξ values

4.2 Effect of ξ on noise tolerance

In order to see the effect of ξ on noise tolerance, we ran a series of experiments to train Hopfield networks varying ξ through a range of values between 0.0 to 1000.0. The noise tolerance of the resulting network was tested by starting the network at each training pattern with a fixed amount of the noise, and the percent of the training patterns which were recalled correxctly was recorded. For each value of ξ, 100 networks were trained, and the mean value and standard deviation of the correct recall percentage are calculated. This average correct recall percentage indicates high noise tolerance.

Table 1, 2, and 3 summarize the results obtained. The tables show the average and standard deviation noise tolerance for 0%, 5%, 10%, 15% and 20%. We varied ξ through five orders of magnitude. In case of $\xi = 0$, the *Pseudo-Relaxation algorithm* is same as the *Relaxation algorithm*. Note that the network always has perfect recall with 0% noise, i.e., the algorithm guarantees 100% storage.

In all cases, it is observed that an improvement in noise tolerance as ξ is increased. However, the noise tolerance does not continue to increase as use arbitrarily large values of ξ . In general, the noise tolerance is optimal when $\xi = 100.0$ with 10 digits training set and the 26 upper- case letters, and the noise tolerance is optimal when $\xi = 10.0$ with the 26 lower-case letters. This observation suggests that there is an intrinsic limit on the best noise tolerance achievable depending on the given set of patterns.

Figures 5, 6, and 7 show the average values of the noise tolerance from the tables.

Table 1 : Effect of ξ on noise tolerance for 10 digits

ξ	0% noise	5% noise	10% noise	15% noise	20% noise
0.0	100.0 ± 0.0	57.8 ± 24.8	35.9± 23.9	23.3 ±22.7	15.9 ± 19.4
0.1	100.0 ± 0.0	62.4 ± 21.9	40.0± 25.5	23.9 ±23.5	17.0 ±18.6
1.0	100.0 ± 0.0	82.4± 12.7	67.9± 21.0	51.4± 22.2	38.1± 23.3
10.0	100.0 ± 0.0	81.8± 17.7	67.9± 23.4	58.3 ±25.2	45.2± 26.1
100.0	100.0 ± 0.0	91.1± 10.4	80.4± 20.2	68.0 ±26.2	57.2 ± 25.90
1000.0	100.0 ± 0.0	87.1 ± 15.3	76.9± 24.7	62.6 ±31.5	52.3± 31.2

Table 2 : Effect of ξ on noise tolerance for 26 upper letters

ξ	0% noise	5% noise	10% noise	15% noise	20% noise
0.0	100.0 ± 0.0	35.9 ± 16.5	15.8 ± 14.7	7.8 ± 10.1	3.9 ± 6.8
0.1	100.0 ± 0.0	39.0 ± 18.7	17.0 ± 15.3	8.8 ± 11.0	4.2 ± 7.2
1.0	100.0 ± 0.0	55.3 ± 18.9	32.4 ± 18.8	21.2 ± 18.9	10.3 ± 11.7
10.0	100.0 ± 0.0	71.7 ± 14.9	51.3 ± 18.3	31.7 ± 17.9	18.5 ± 13.1
100.0	100.0 ± 0.0	70.7 ± 17.1	51.6 ± 18.9	34.8 ± 17.1	22.1 ± 14.4
1000.0	100.0 ± 0.0	70.4 ± 15.1	48.4 ± 18.0	32.7 ± 16.0	20.1 ± 14.1

Table 3 : Effect of ξ on noise tolerance for 26 lower letters

ξ	0% noise	5% noise	10% noise	15% noise	20% noise
0.0	100.0 ± 0.0	31.9 ± 15.7	12.3 ± 11.6	5.7 ± 7.9	2.4 ± 4.0
0.1	100.0 ± 0.0	34.8 ± 17.6	16.0 ± 13.9	7.2 ± 8.5	3.2 ± 4.9
1.0	100.0 ± 0.0	52.9 ± 16.8	28.3 ± 14.9	16.5 ± 10.7	7.5 ± 6.5
10.0	100.0 ± 0.0	64.5 ± 19.0	42.3 ± 20.4	25.3 ± 17.9	14.7 ± 13.1
100.0	100.0 ± 0.0	59.5 ± 17.5	38.0 ± 21.9	22.2 ± 19.1	13.5 ± 15.2
1000.0	100.0 ± 0.0	67.0 ± 17.3	43.8 ± 19.1	26.5 ± 17.5	15.0 ± 13.5

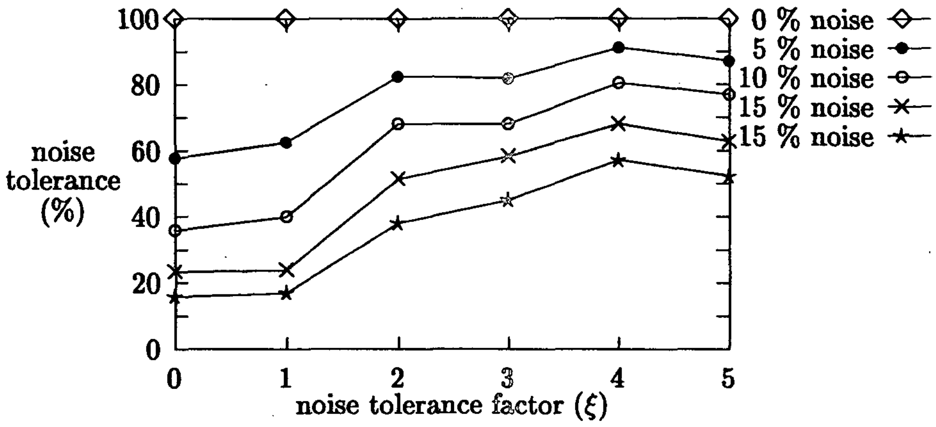

Figure 5: Effect of ξ on noise tolerance for 10 digits

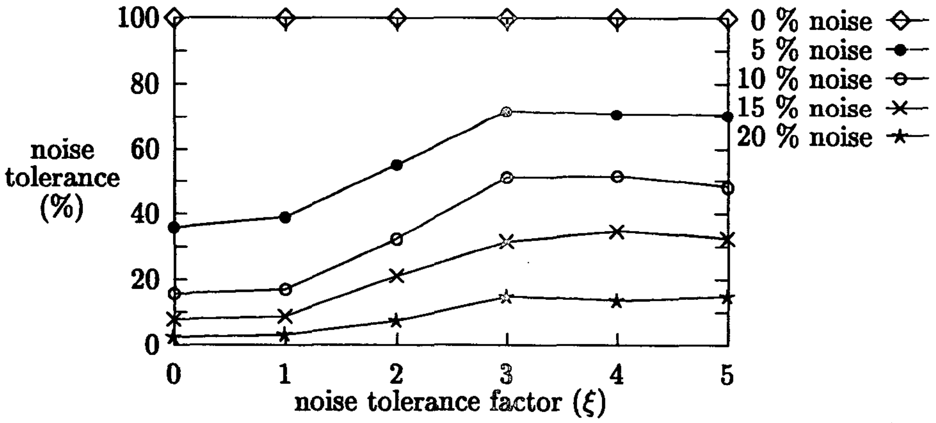

Figure 6: Effect of ξ on noise tolerance for 26 upper letters

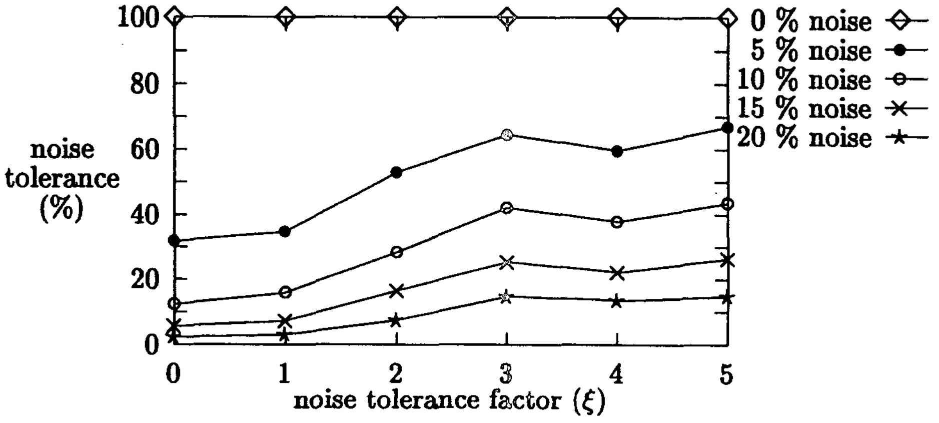

Figure 7: Effect of ξ on noise tolerance for 26 lower letters

5. Conclusion

After adding a noise tolerance factor to the *Relaxation* learning algorithm in Hopfield network, the noise tolerance is significantly improved. This algorithm with the noise tolerance factor is called the *Pseudo-Relaxtion* algorithm, and it has been proved effectively. Unlikely unlearning techniques coupled with variations of Hebian learning, the noise tolerance factor does not effect the storage capacity.

The effect of the noise tolerance factor to improve the noise tolerance is analyzed geometrically, and serveral experiments are made to study the noise tolerance factor in Hopfield network. Three character sets from IBM CGA font are used as training samples. The experimental study also shows that the noise tolerance factor does not effect the learning speed.

References

1. Motzkin, T. S. and Schoenberg, N. J. (1954). The Relaxation Method for Linear Inequalities, *Canadian J, Math*, Vol. 6, No. 3, 355-357

2. Nilsson, N. J. (1990). *The Mathematical Foundations of Learning Machines*, Morgan Kaufmann Publishers, San Mateo, Califormia.

3. Oh, H. and Kothari, S. C. (1992). A Pseudo-Relaxation Learning Algorithm for Bidirectional Associative Memory, *IEEE Int. Joint Conf. on Neural Networks*, Baltimore.

4. Wong, A. J. (1988). Reconginition of general patterns using neural networks, *Biological Cybernetics*, 58, 361-372.