1998 International Computer Symposium
Workshop on Artificial Intelligence
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

A HYBRID PARADIGM FOR WEIGHT INITIALIZATION IN SUPERVISED FEEDFORWARD NEURAL NETWORK LEARNING

Leandro Nunes de Castro, and Fernando José Von Zuben

Department of Computer Engineering and Industrial Automation (DCA)
School of Electrical and Computer Engineering (FEEC)
State University of Campinas, Campinas, Brazil
E-mail: {lnunes, vonzuben}@ dca.fee.unicamp.br

ABSTRACT

The initial set of weights to be used in supervised learning for multilayer neural networks has a strong influence in the learning speed and in the quality of the solution obtained after convergence. An inadequate initial choice for the weight values may cause the training process to get stuck in a poor local minimum or to face abnormal numerical problems. Nowadays, there are several proposed techniques that try to avoid both local minima and numerical instability, only by means of a proper definition of the initial set of weights. However, the problem of the majority of these approaches is that they persist on ignoring useful properties of the training set when presented to the neural network. An alternative hybrid paradigm for weight initialization in feedforward neural networks is proposed here, and applied to several benchmark problems. The results are then compared with that produced by other approaches found in the literature.

1. INTRODUCTION

The efficacy and efficiency of supervised learning in multilayer neural networks strongly depends on:

- the network topology;
- the neurons' activation function;
- the learning rule;
- the initial values of the weights.

Optimal instances for these items are usually unknown *a priori* because they depend mainly on the particular training set to be considered and on the nature of the solution [16].

Here we assume that the network topology, the neurons' activation function and the learning rule have already been determined in a proper manner, though not necessarily the optimal one. Under these conditions, a successful training process turns to depend solely on a good instantiation of the set of weights, that is, one that guides the training process to a high-quality solution, out of poor local minima and abnormal numerical problems.

At this point, it is well known that supervised learning can be viewed as an optimization problem, with the cost function being a function of the set of weights, also known as the parameters of the neural network [5].

The importance of a good choice for the initial set of weights is stressed by Kolen and Pollak [8]. They showed that it is not feasible to perform a global search for obtaining the optimal set of weights. So, for practical purposes, the learning rule should be based on optimization techniques that employ local search to find the optimal solution [14]. But local search implies that the solution has a strong relation to the initial condition, because each initial condition belongs to the basis of attraction of a particular local minimum, which will attract the solution [6].

Consequently, only local minima can be produced in practice as the result of a well-succeeded training process. If such a minimum happens to be the global minimum or a good local minimum of the cost function, the result is a properly trained neural network. Otherwise, an inferior result will be achieved, so that the poorer the local minimum, the worse the performance of the trained neural network.

2. TWO ALTERNATIVE PARADIGMS FOR WEIGHT INITIALIZATION

Several weight initialization techniques for multilayer networks have already been suggested. The simplest methods among them are based on uniform random distribution [8], representing the complete absence of knowledge about the training data set. Concerning more refined approaches, there are basically

two alternative paradigms to search for good initial weights' set in supervised learning, i.e., good initial condition for the optimization process:

- easiest-path paradigm: is not so common in literature. The main idea is to find an initial condition not necessarily close to the optimal solution, but such that the learning process can evolve faster in average, and more efficiently, from the initial condition. The simplest strategy is to automatically define a proper initial interval for the weights and to use a uniform distribution over this interval.
- shortest-path paradigm: is the approach generally employed in the literature. The basic idea is to search for an initial condition as close as possible to the optimal solution, still unknown. The intuitive idea behind this approach is that the closer the initial condition to the optimal solution, the less probable is the appearance of a poor local minimum in the path between, and the more efficient becomes the training process. Two strategies can be considered: knowledge extraction from the training set to discover peculiarities of the optimization surface (based on theoretical aspects), or sampling exploration of the optimization surface to improve the chance to find a promising region to search for the optimum (based on empirical aspects).

The easiest-path paradigm generally neglects the training set in the attempt to define a good range of values for the weights. As a consequence, the path between the initial condition and the optimal solution, tough easy to go through, can be very long.

On the other hand, the shortest-path paradigm takes into account the whole training set, but generally neglects the consequences of the combination ⟨input-output data⟩ + ⟨weights⟩ in the neural network signal processing. As a consequence, the path between the initial condition and the optimal solution, tough short, can be very hard to go through.

In section 5 we will propose a hybrid paradigm that represents a compromise between the shortest and the easiest paths in the hope of achieving a more robust and efficient initialization technique.

3. AN EXAMPLE OF THE EASIEST-PATH PARADIGM

Fahlman [5] performed studies about random weight initialization techniques for multilayer neural networks. He proposed the use of a uniform distribution over the interval [–1.0, 1.0], but experimental results showed that the best initialization interval to the problems he dealt with varied in ranges between [–0.5, 0.5] and [–4.0, 4.0].

Some researchers tried to determine the best initialization interval using other neural network parameters.

Define a as a small real value and d_{in} the neuron *fan-in*. Boers and Kuiper [2] initialize the weights using a uniform distribution over the interval $\left[-3/\sqrt{d_{in}}, 3/\sqrt{d_{in}}\right]$, without any mathematical justification.

Nguyen and Widrow [12] proposed a simple modification of the random initialization process. Their approach is based on a geometrical analysis of the hidden units' response to a single input. This analysis can be extended to multiple input using Fourier transform. The weights connecting the output units to the hidden units are initialized with small random values over the interval [–0.5, 0.5]. The initial weights at the first layer are designed to improve the learning capabilities of the hidden units. Using a scale factor, $\beta = 0.7(q)^{1/p}$, where q is the number of hidden units and p is the number of inputs, the weights are randomly initialized and then scaled by $\mathbf{v} = \beta \frac{\mathbf{v}}{\|\mathbf{v}\|}$, where $\mathbf{v}$ is the first layer weight vector.

Kim and Ra [7] calculated a lower bound for the initial length of the weight vector of a neuron to be $\sqrt{\alpha/d_{in}}$, where α is the learning rate.

4. AN EXAMPLE OF THE SHORTEST-PATH PARADIGM

Lehtokangas *et. al.* [9] proposed a method based upon the orthogonal least squares (OLS) algorithm. The OLS algorithm is being widely and successfully used in the training of radial basis function network (RBF) [4].

4.1. The OLS algorithm

A multilayer neural network architecture can be considered as a regression model where the hidden units are the regressors. In the weight initialization process, the problem is to determine which are the best regressors available. An efficient algorithm to determine the optimal regressor is the OLS algorithm [9].

Define p and q as the number of input and hidden units respectively. Let s be the desired output of the net. A regression model for this output can be given, in matrix notation, by:

$$\mathbf{s} = \mathbf{R}\theta + \varepsilon, \tag{1}$$

where $\mathbf{s}^T = [s^1, s^2, ..., s^n]$, $\mathbf{R}$ is the regression matrix (fixed functions of the inputs), $\theta^T = [\theta_0, \theta_1, ..., \theta_M]$ are the parameters of the model, $\varepsilon^T = [\varepsilon^1, \varepsilon^2, ..., \varepsilon^n]$ are the sampling noise vector, and M is the number of regressors. The regression matrix $\mathbf{R}$

$$\mathbf{R} = \begin{bmatrix} 1 & R_1[\mathbf{x}^1] & \cdots & R_M[\mathbf{x}^1] \\ \cdots & \cdots & \cdots & \cdots \\ 1 & R_1[\mathbf{x}^n] & \cdots & R_M[\mathbf{x}^n] \end{bmatrix}, \tag{2}$$

can be decomposed into

$$\mathbf{R} = \mathbf{HU}, \tag{3}$$

where the above equation represents an orthogonal decomposition, with $\mathbf{U}(M+1)\times(M+1)$ upper triangular with 1's on its diagonal.

$$\mathbf{U} = \begin{bmatrix} 1 & \alpha_{12} & \alpha_{13} & \cdots & \alpha_{1(M+1)} \\ 0 & 1 & \alpha_{23} & \cdots & \alpha_{2(M+1)} \\ 0 & 0 & 1 & \cdots & \alpha_{3(M+1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4}$$

and $\mathbf{H}$ is a $n \times (M+1)$ matrix with orthogonal columns $\mathbf{h}_j$ such that:

$$\mathbf{H}^T\mathbf{H} = \mathbf{B} \tag{5}$$

The matrix $\mathbf{B}$ is diagonal and its elements are:

$$b_{jj} = \mathbf{h}_j^T\mathbf{h}_j = \sum_{i=1}^{n} (h_j^i)^2, \quad j = 1,2,...,M+1. \tag{6}$$

When the condition $\mathbf{U}\theta = \mathbf{g}$ is satisfied, equation (1) can be written:

$$\mathbf{s} = \mathbf{Hg} + \varepsilon. \tag{7}$$

The least squares estimate for the new parameter vector $\mathbf{g}$ can be calculated by the expression:

$$\hat{\mathbf{g}} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{s} = \mathbf{B}^{-1}\mathbf{H}^T\mathbf{s}. \tag{8}$$

Now, the orthogonal decomposition of equation (3) can be obtained using the Grahm-Schimidt procedure

$$\left.\begin{aligned} \mathbf{h}_1 &= \mathbf{r}_1 \\ \alpha_{ij} &= \frac{\mathbf{h}_j^T\mathbf{r}_i}{\mathbf{h}_j^T\mathbf{h}_j} \\ \mathbf{h}_j &= \mathbf{r}_j - \sum_{i=1}^{j-1}\alpha_{ij}\mathbf{h}_i \end{aligned}\right\} \begin{matrix} i = 1,2,...,(j-1) \\ j = 2,3,...,(M-1) \end{matrix}. \tag{9}$$

The regressors $\mathbf{h}_i$ and $\mathbf{h}_j$ are orthogonal for $i \neq j$, and the squared sum is given by:

$$\mathbf{s}^T\mathbf{s} = \sum_{j=1}^{M+1} g_j^2\mathbf{h}_j^T\mathbf{h}_j + \varepsilon^T\varepsilon. \tag{10}$$

If $\mathbf{s}$ is the desired output vector after its mean has been removed, then its variance is estimated as:

$$\text{var}(\mathbf{s}) = \frac{1}{n}\mathbf{s}^T\mathbf{s}. \tag{11}$$

The term summing equation (1) is the desired output variance part that can be explained by the regressors $\mathbf{h}_j$. However, each regressor has its own contribution to the total sum and the problem is to determine the q greatest contribution regressors. An error reduction ratio can be defined as:

$$err_j = \frac{g_j^2\mathbf{h}_j^T\mathbf{h}_j}{\mathbf{s}^T\mathbf{s}}, \quad j = 1,2,...,(M+1) \tag{12}$$

The practical procedure can be loosely described as follows:

- calculate the error reduction ratio for each of the original regressors ($\mathbf{h}_j = \mathbf{r}_j$) and select the one with the largest ratio. The regressor selected becomes $\mathbf{h}_1$ and is taken from the $\mathbf{r}_j$ set;
- use the $\mathbf{r}_j$ remaining regressors as the candidates for obtaining $\mathbf{h}_2$;
- repeat the previous steps until the q best regressors have been selected.

4.2. Initializing weights with the OLS algorithm

Consider a linear output net, and the hyperbolic tangent (*tanh*) as the hidden units' activation function. Let $\mathbf{X}$ be the data matrix and $\mathbf{v}$ the first layer weight vector. The regression matrix is given by

$$\mathbf{R} = \tanh(\mathbf{v}^T\mathbf{X}), \quad \text{or}$$

$$R_j = \tanh\left(\sum_{i=1}^{p} v_{ij}x_i\right) \tag{13}$$

It must be observed that, during the initialization process, the number of hidden units is M, and must be significantly larger than the desired number q (for example, $M = 10q$).

Equation (13) shows that each of the M hidden units corresponds to a regressor, then we must use the OLS algorithm to select the q best. Before using the OLS algorithm, the M regressor candidates must be generated. One simple way of initializing the candidate weights is using a uniform distribution over an interval $[-a, a]$. If a regressor is chosen by the OLS algorithm, then the

initial values of the selected regressors are the initial values of the neural network weights. Each regressor has $p + 1$ connections. The number of inputs determines the dimension of the weight space formed by the regressors. The smaller the dimension of the weight space, the fewer degrees of freedom exist to initialize the regressors.

The initialization phase can be summarized as follows:

- linearize the activation function of the output neuron. Define M regressors such as $M >> q$. Initialize the regressors, i.e., the first layer weight vector v with uniformly distributed values. Select the q best regressors using the OLS algorithm and let the initial values of the selected regressors be the initial values for the network.
- calculate the outputs of the q previously selected hidden units for each training pattern. Form a linear regression for the linear part of the network and let the obtained regression coefficients be the initial values for the weight vector w of the last (second) layer.

5. A HYBRID PARADIGM

We now propose a simple and efficient paradigm that can be interpreted as something in between the easiest and the shortest-path paradigm. This hybrid paradigm explores the information contained in the training data set at the same time that it tries to consider the signal processing aspects throughout the neural network.

It is well-known that, once saturated, every neural network node cannot easily escape from this operation point, because its gradient[1] of error is so small that the magnitude of the search direction for its weights is almost zero, and the corresponding weights will not significantly change in future iterations. Stäger and Agarwal [15] developed an algorithm for detecting "saturated" hidden nodes and somehow re-activating them while transferring their contribution onto the bias node at the same layer.

In our approach, we try to avoid hidden node saturation at the moment of weight initialization, by means of properly defining the initial weight values. To do that, we consider both the training data set and its effect over the layer-by-layer processing.

The goal is to guarantee that, for the input data under consideration, the hidden nodes will be initially active exclusively in the approximately linear part of the activation function. This idea makes the algorithm more flexible to run faster into the direction of the optimal solution, just as the policy of the easiest-path paradigm, and the range for weight definition is obtained here as a function of the learning data set, just as the policy of the shortest-path paradigm, and with a low computational cost.

To develop our method, let's consider that the data has a uniform or gaussian distribution around a mean. Other distributions could be considered instead, without loss of generality. In Figure 1, we depict our goal by plotting the data distribution around the approximately linear part of the activation function. Figures 2 and 3 depict cases we try to avoid. We do not want the initial weights to be set out of the approximately linear part of the activation function, as shown in Figure 2 for some subset of the training data, and in Figure 3 for all data set. It is clear that if we initialize the weights such that the internal response of one or more nodes turn to be out of the linear part of the neural network, the derivative of the activation function will be approximately zero, making the training process much slower and subject to numerical problems.

Let v be the input to hidden layer weight vector, X the training data set and z the output of the hidden nodes when X is presented to the neural net.

In matrix notation, the output z of the hidden nodes can be calculated by the expression

$$\mathbf{z} = \tanh(\mathbf{v}^T\mathbf{X}). \qquad (14)$$

If the hidden nodes are supposed to respond in the linear part of their activation functions, then equation (14) can be rewritten by the approximate form:

$$\mathbf{z} = \mathbf{v}^T\mathbf{X}. \qquad (15)$$

Now the following question can be raised: "Given any multiple input distribution, which is the best linear combination of these inputs such that the output combination z is a normal distribution of zero mean and small variance?"

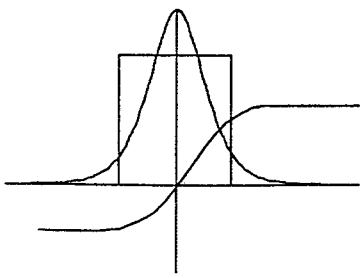

Figure 1: Hyperbolic tangent activation function, with data distribution (uniform or gaussian) within the approximately linear part of the function.

[1] Depending on the precision of the digital computer used, the gradient eventually evaluates to zero.

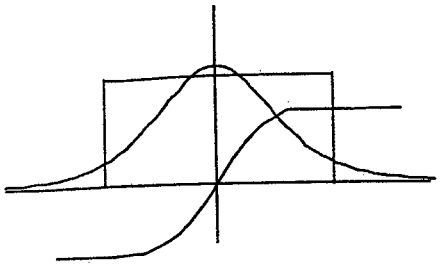

Figure 2: Hyperbolic tangent activation function, with data distribution (uniform or gaussian) partially out of the approximately linear part of the function.

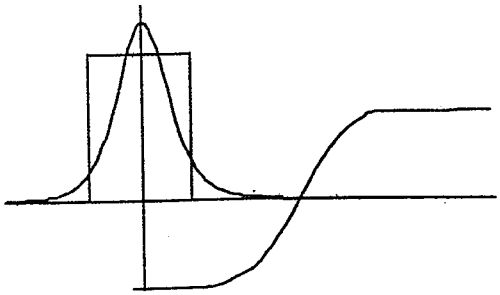

Figure 3: Hyperbolic tangent activation function, with data distribution (uniform or gaussian) completely out of the approximately linear part of the function.

The answer to this question is the solution to the following minimization problem:

$$\min_{\mathbf{v}\neq\mathbf{0}} \left\| \mathbf{v}^T\mathbf{X} - \mathbf{z} \right\|. \qquad (16)$$

The restriction imposed on the minimization problem of equation (16) is due to the fact that we are obviously not interested in the trivial solution, i.e., $\mathbf{v} = \mathbf{0}$. We can avoid the trivial solution by adopting a uniform distribution with zero mean and fixed variance for $\mathbf{z}$, say $\overline{\mathbf{z}}$.

A solution to the problem presented in equation (16) is obtained by means of pseudo-inversion:

$$\mathbf{v} = \left(\mathbf{X}\mathbf{X}^T\right)^{-1}\mathbf{X}.\overline{\mathbf{z}}^T. \qquad (17)$$

In this approach, we must guarantee that $\mathbf{X}$ is a full rank matrix, what can be easily achieved by simply manipulating the training data to avoid redundancy.

As the last step of our proposed algorithm, we need to initialize the weights at the subsequent layers. We initialize them all with the same small value, instead of using a normal or uniform random distribution, because the output of the first hidden layer is already characterized by uniform or gaussian distribution around zero, that we have to preserve until the output layer.

6. ALGORITHMS AND BENCHMARKS

We compare our algorithm performance with four other methods applied to seven benchmark problems. The methods compared were BOERS [2], WIDROW [12], KIM [7], OLS [9], and INIT, where INIT is the method we propose and the others were described in Sections 3 and 4.

To specify the benchmark problems used, let N be the number of samples, SSE the desired sum squared error (stopping criterion) and *net* the net architecture represented by $[n_i\text{-}n_h\text{-}n_o]$. Where n_i is the number of inputs, n_h is the number of hidden units and n_o is the number of outputs of the network

The benchmarks used for comparison were:

- parity 2 problem (XOR): $N = 4$, *net*: [2-2-1], SSE = 0.01;
- parity 3 problem: $N = 8$, *net*: [3-3-1], SSE = 0.01;
- $\sin(x).\cos(2x)$: $N = 25$, *net*: [1-10-1], SSE = 0.1;
- ESP: real world problem used by Barreiros *et. al.* [1]; $N = 75$, *net*: [3-10-5], SSE = 0.1;
- SOYA: another real world problem used by Castro *et. al.* [3], $N = 116$, *net*: [36-10-1], SSE = 0.1;
- IRIS: this benchmark is part of the machine learning database and is available in [11]; $N = 150$, *net*: [4-10-3], SSE = 0.15; and
- ENC/DEC: the family of encoder/decoder problem is very popular and is described in [5]. $N = 10$, *net*: [10-7-10].

The training algorithm used was the Moller scaled conjugate gradient [10], with the exact calculation of the second order information [13]. The vector $\overline{\mathbf{z}}$ was initialized with a uniform distribution over the interval [-1.0, 1.0] and $\mathbf{w} = \mathbf{0.1}$ (vector containing only terms with the value 0.1).

For the encoder/decoder problem, the algorithms were not able to converge for all runs. In order to report these results, we performed 20 runs in this case and present the number of successes (convergence) achieved by each algorithm.

7. SIMULATION RESULTS

For each algorithm and each benchmark problem we performed 10 runs. The results to be presented will be the minimum, maximum, mean and standard deviation of the number of epochs necessary for convergence.

It can be seen, in Tables 1 and 2 (see Appendix), that the method proposed presents a superior performance in the majority of the cases tested.

Figure 4 (see Appendix) makes a comparison of the performance behavior of the algorithms tested. One algorithm is considered to be superior to another when its maximum, minimum, mean and standard deviation number of epochs for convergence is the smallest.

8. CONCLUSIONS

The results showed that the proposed hybrid paradigm for weight initialization is superior on average to the other pure paradigms when tested in artificial and real-world problems.

In Table 1, we can see that OLS algorithm only presents the smallest number of epochs for convergence when applied to some artificial problems. On the other hand, its behavior is very poor in the majority of real world applications tested (SOYA and IRIS).

After the method proposed in this paper, the one that presented the second best general performance was the OLS strategy. It is also important to notice that the OLS algorithm has the disadvantage of being very time consuming, due to the search among candidates, the application of OLS algorithm and the linear regression in the output layer.

REFERENCES

[1] **Barreiros, J.A.L., Ribeiro, R.R.P., Affonso, C.M. & Santos, E.P.**, "Estabilizador de Sistemas de Potência Adaptativo com Pré-Programação de Parâmetros e Rede Neural Artificial", 1996.

[2] **Boers, E.G.W. & Kuiper, H.** "Biological Metaphors and the Design of Modular Artificial Neural Networks", Master Thesis, Leiden University, Leiden, Netherlands, 1992.

[3] **Castro, L.N., Von Zuben, F.J. & Martins, W.** "Hybrid and Constructive Neural Networks Applied to a Prediction Problem in Agriculture". *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 3, pp. 1932-1936, May 1998.

[4] **Chen, S., Chung, E.S. & Alkadhimi, K.** "Regularised Orthogonal Least Squares Algorithm for Constructing Radial Basis Function Networks", *Int. J. Control*, v. 64, n° 5, pp.829-837, 1996.

[5] **Fahlman, S., E.**, "An Empirical Study of Learning Speed in Back-Propagation Networks", *Tech. Rep.*, CMU-CS-88-162, School of Computer Science, Carnegie Mellon University, Pittsburg, PA, September 1988.

[6] **Hertz, J., Krogh, A. & Palmer, R.G.** Introduction to the Theory of Neural Computation. Addison-Wesley Publishing Company, 1991.

[7] **Kim, Y.K. & Ra, J.B.** "Weight Value Initialization for Improving Training Speed in the Backpropagation Network", *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 3, pp. 2396-2401, 1991.

[8] **Kolen, J.F. & Pollack, J.B.** "Back Propagation is Sensitive to Initial Conditions", *Technical Report TR 90-JK-BPSIC*, 1990.

[9] **Lehtokangas, M., Saarinen, J., Kaski, K. & Huuhtanen, P.** "Initializing Weights of a Multilayer Perceptron by Using the Orthogonal Least Squares Algorithm", *Neural Computation*, vol. 7, pp. 982-999, 1995.

[10] **Moller, M.F.** "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning", *Neural Networks*, vol. 6, pp. 525-533, 1993

[11] Neural Networks Benchmarks (UCI machine learning databases): ftp://ftp.ics.uci.edu/pub/machine-leraning-databases.

[12] **Nguyen, D. & Widrow, B.** "Improving the Learning Speed of two-layer Neural Networks by Choosing Initial Values of the Adaptive Weights", in *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, Ann Arbor, MI, vol. 3, pp. 21-26, 1990.

[13] **Pearlmutter, B.A.** "Fast Exact Calculation by the Hessian", *Neural Computation*, vol. 6, pp. 147-160, 1994.

[14] **Shepherd, A.J.** "Second-Order Methods for Neural Networks – Fast and reliable Methods for Multi-Layer Perceptrons", Springer, 1997.

[15] **Stäger, F. & Agarwal, M.** "Three Methods to Speed up the Training of Feedforward and Feedback Perceptrons", *Neural Networks*, vol. 10, n° 8, pp. 1435-1443, 1997.

[16] **Thimm, G. & Fiesler, E.** "High-Order and Multilayer Perceptron Initialization", *IEEE Trans. on Neural Networks*, vol. 8, n° 2, pp. 349-359, 1997.

APPENDIX

Table 1: Simulation results for the benchmarks considered and methods tested. *Max*, *min* and *mean* are the maximum, minimum and mean number of epochs, respectively. δ is the standard deviation of the number of epochs.

Problem	Method	*Max*	*Min*	*Mean*	δ
parity 2	BOERS	129	6	42.90	48.32
	WIDROW	93	8	20.50	25.92
	KIM	84	6	32.50	25.91
	OLS	166	5	46.70	46.50
	INIT	47	8	18.60	12.76
parity 3	BOERS	62	10	25.10	16.01
	WIDROW	25	10	19.60	5.17
	KIM	46	10	18.90	10.41
	OLS	148	29	65.70	36.16
	INIT	24	8	16.00	4.97
$\sin(x).\cos(2x)$	BOERS	187	79	135.10	35.39
	WIDROW	243	123	178.60	41.27
	KIM	231	95	164.50	44.89
	OLS	133	39	91.40	33.06
	INIT	449	181	254.80	80.69
ESP	BOERS	1618	340	883.40	467.52
	WIDROW	2280	236	825.40	639.74
	KIM	1763	425	715.40	397.82
	OLS	2052	35	545.42	586.62
	INIT	762	383	479.20	118.19
SOYA	BOERS	174	136	158.60	13.14
	WIDROW	464	176	266.30	79.27
	KIM	280	177	219.40	28.34
	OLS	5000	4303	4922.40	219.05
	INIT	236	136	188.00	29.30
IRIS	BOERS	1568	735	1102.40	281.33
	WIDROW	1240	676	918.60	183.72
	KIM	2063	767	1294.80	438.34
	OLS	5000	2075	4140.20	1384.74
	INIT	1407	662	869.80	216.73

Table 2: Encoder/decoder (ENC/DEC) problem, convergence capabilities.

Problem	Method	*Max*	*Min*	*Mean*	δ	*Convergence*
ENC/DEC	BOERS	40	12	22.21	7.94	14/20
	WIDROW	32	7	12.36	6.81	14/20
	KIM	298	10	38.17	69.07	18/20
	OLS	276	7	97.78	102.03	9/20
	INIT	16	6	9.44	2.53	16/20

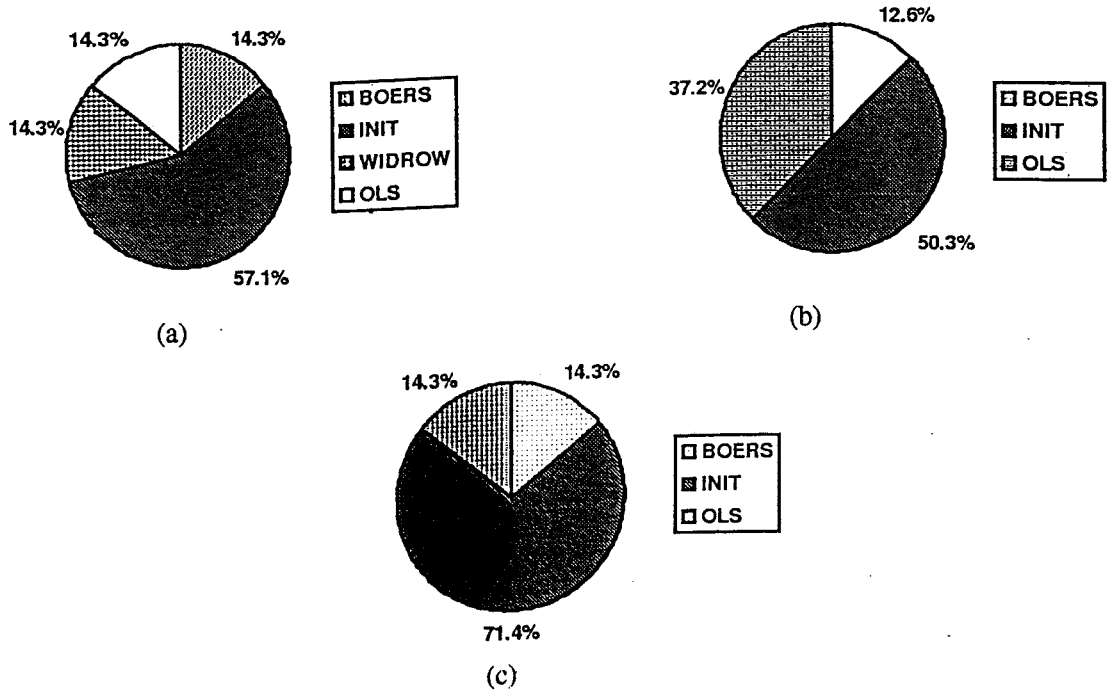

Figure 4: Performance comparison of the methods. (a) In 57.1% of the cases, INIT needs the smallest maximum number of epochs for convergence and has the smallest standard deviation. (b) In 50.3% of the cases, INIT needs the smallest number of epochs for convergence. (c) In 71.4% of the cases, INIT method presents the smallest mean number of epochs for convergence.