Cluster Learning for the Approximation of Dynamical System

M. CONTI, S. ORCIONI, C. TURCHETTI
Department of Electronics
University of Ancona
via Brecce Bianche, 60131 Ancona
ITALY

Abstract: - An improved stochastic learning algorithm, called Cluster Random Weight Change (CRWC), is applied in this work to train a Neural Network in the approximation of static and dymamical systems. The network is subdivided in a fixed number of neurons to form some clusters of neurons and learning is applied to one cluster at a time. The algorithm has been implemented in a PC and used to train an analog Neural Network chip in a "chip in the loop" learning scheme. The stochastic algorithm has been modified with respect to a previous version to improve its learning accuracy and to reduce the critical dependence on noise amplitude that is a drawback of stochastic learning algorithms.

Key-Words:Neural Networks, Stochastic, Learning, Dynamical Systems *CSCC'99 Proceedings, Pages:3021-3025*

1 Introduction

The introduction of noise during supervised learning allows the neural network to reach the global optimum avoiding local minima in the error function. Some examples of stochastic learning are: simulated annealing, backpropagation with weight perturbation, introduction of weight noise during training[1], Random Weight Change [2]. Furthermore it has been shown that training with noise may improve the generalization capability of the network and reduce the negative effects of the weight variations after learning [2-3].

On the other hand, stochastic learning algorithms show a critical dependence on the parameters of the annealing function, that is the function which reduces the noise amplitude during learning.

To solve this problem the Cluster Random Weight Change (CRWC) learning algorithm has been introduced [2]. In this work CRWC has been modified to improve learning accuracy and stability, to reduce the dependence on noise amplitude and to reduce the number of iterations required to reach the optimum. The algorithm has been applied to train an analog neural network chip in the approximation of static and dynamic systems.

2 Cluster Random Weight Change

The network is subdivided in a fixed number of neurons to form some groups or clusters of neurons, as shown in Fig. 1. Learning is applied to one cluster at a time for some steps, keeping the other weights fixed to the value they reached at the end of their learning time. This cyclic procedure is repeated until the error is acceptable. The error is calculated as the difference between the desired output and the output of the complete network.

The condition used in [2] to switch learning from one cluster to another is related to the number of iterations dedicated to each cluster. The criterion is very simple and can be easily implemented in hardware. An improvement in the performances of the algorithm has been reached by adding some conditions on the variation of the error during the learning of each cluster. These conditions are based on the considerations that: i) a cluster may be incapable of decreasing the error due to the values of the weights of the other clusters (in this case further iterations in training this cluster are useless); ii) the stability of the algorithm is improved if only small variations of the error are allowed for each cluster.

For this reason the following two variations of the algorithm have been tested.

A) Learning switches between one cluster to another when at least one of the following conditions is true:
   - the number of iterations for the current cluster is greater than a fixed number I_{max};
   - from the beginning of the training of the current cluster the increment of the error is greater than a fixed value $\Delta E+$;
   - from the beginning of the training of the current cluster the decrement of the error is greater than a fixed value $\Delta E-$.

B) Learning switches between one cluster to another when at least one of the following conditions is true:
   - the number of iterations for the current cluster is greater than a fixed number I_{max};
   - from the beginning of the training of the current cluster the increment of the error is greater than a fixed value $\Delta E+$;
   - the error function does not decreases for more than a fixed value I_{Δ} of iterations.

For comparison the criterion used in [2] is reported:

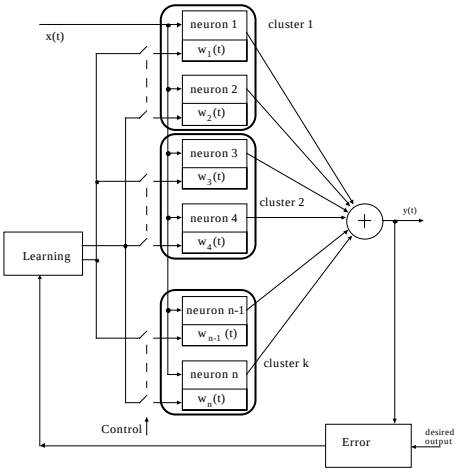


Fig.1 Example of Cluster Network partition

C) Learning switches between one cluster to another when the number of iterations for the current cluster is greater than a fixed number I_{max}.

Each cluster is trained on turn using Random Weight Change (RWC) algorithm [2] which is based on the well known Brownian motion equation, defined by the following discrete time algorithm:

```
if [ E(w(i))>E(w(i-1) ]
  then v(i+1)=v(k)+η[n(i)-v(i)]
  else v(i+1)=v(i)
end if
w(i+1)=w(i)+μ v(i+1)
```

where i is the iteration number, w is the weight vector, and n(i) is a random process vector, with standard deviation vanishing as the iteration number increases or the error decreases, E(w(i)) is the error, η and μ are learning parameters. In the following a brief description of the algorithm used is reported:

```
m=0
while ( E < E_desired and  m < itermax)
     for j=1 to k  /* for each cluster */
        i=0
        E_average=0
        while ( swith criterion )
           calculate error E
           E_average=E+E_average
           update j-th cluster of using RWC
           m=m+1
           i=i+1
        end while
        E_average=E_average/iter
        update noise variance σ_n^2(E_average,m)
     end for
end while
```

Where k is the number of clusters, M the number of weights for each cluster, N=k*M the total number of weights, iter is the number of iteration for each cluster.

The proposed algorithm is general and can be applied to many optimization problems such as learning of different types of Neural Networks. In the following examples the algorithm has been applied to a class of Neural Networks called Approximate Identity Neural Networks (AINN)[4] whose input-output relationship is:

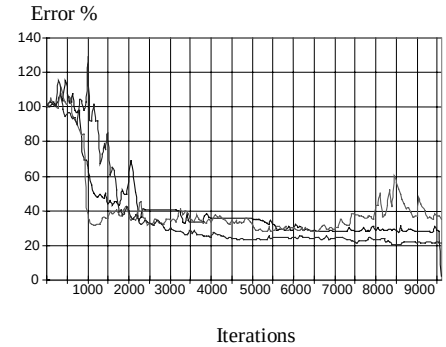


Fig. 2a Neural network training with CRWC and switch criterion A (I_{max}=100, ΔE-=0.3, ΔE+=0)

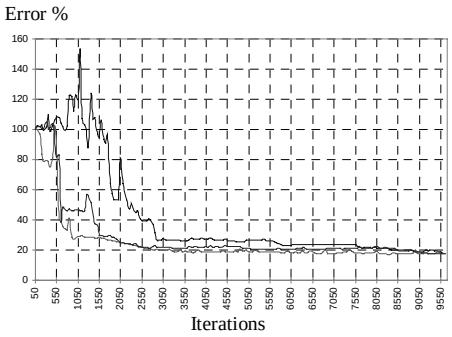


Fig. 2b Neural network training with CRWC and switch criterion B (I_{max}=100, I_{Δ}=50, ΔE+=0.6)

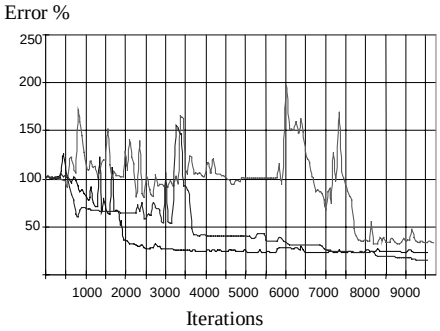


Fig. 2c Neural Net. training with CRWC and switch criterion C (I_{max}=50)

$$\Lambda_n(x;w) = \sum_{j=1}^{n} c_j \Omega_j = \tag{1}$$

$$\sum_{j=1}^{n} c_j \sum_{i=1}^{p} \left\{ \tanh\left(\frac{n_{ij}(x_i - t_i) + \sigma_{ij}}{2} \right) - \tanh\left(\frac{n_{ij}(x_i - t_i) - \sigma_{ij}}{2} \right) \right\}$$

where $w=\{n,t,\sigma\} \in R^r$ is the weight vector, $x \in R^p$ the input vector. The first example reported is the training of an AINN with a one dimensional input. The simulations have been performed with matlab. The desired functions is $f = 0.6sin(\pi x)$. Figs. 2a-c reports the error during learning for the three (A-B-C) switching criteria of CRWC. Three different random sequences are reported in each figure.

Learning speed is higher with algorithms A and B. The results show an increment in learning accuracy (the error is often lower for cluster learning A and B) and a less critical dependence on the amplitude of noise and on the annealing parameters.

3 Experimental Results: Off-Chip Learning

3.1 Function Approximation.

Cluster Random Weight Change has been tested by a chip inserted in the learning loop scheme as shown in Fig.3.

We used a one-input one-output Neural Network made of 4 chips with 6 neurons each fabricated at the IRST-laboratory (Trento-Italy) with a 2μm CMOS technology [5]. The communication between the host-PC, which performs learning, and the neural network has been achieved by means of an interface on AT bus with 16 output and 8 input analog channels.

Figs 4-8 show the experimental results obtained training the analog Neural Network as reported in the schematic of Fig.2 by using CRWC (learning algorithm B).

Figs 4a-4d are photos of the oscilloscope in different time instants during learning. The Figures show the output of the Neural Network and the desired output which is the function $y = 1-\exp(-x/\tau)$. Fig. 5 reports the error during. Similar results are reported in Figs. 6a-d and 7, in this case the desired function is function $y = \sin(\pi x)/x$.

Figs 8a-8d show the desired output function $y=x^3$ (continuous lines) and the output of the neural network (dots) during learning: iteration 1800 for Fig. 8.a, 2700 for Fig. 8.b, 7200 for Fig. 8.c and 1000 for Fig. 8.d.

3.2 Trajectory Approximation.

It has been demonstrated that artificial neural networks are able to approximate arbitrary continuous input-output mappings. Similar results has been established with regards to the approximation of some classes of continuous time dynamical systems [6]. In [7] it is shown that, under wide conditions, the trajectory of a dynamical system expressed as ordinary differential equations can be approximated with an arbitrary error by continuous-time recurrent AINN, provided that the initial conditions of the two systems are close to each other.

In this work CRWC learning algorithm has been used to approximate the trajectories of a dynamical system.

The analog neural network chips used in the function approximation described in previous section has been used to implement the Approximating Dynamical System described by the following differential equation.

$$\begin{cases} \dot{y}_1 = a y_2 \\ \dot{y}_2 = \Lambda_n(y_2; w) - b y_2 + A\cos\omega t \end{cases} \quad (2)$$

where $\Lambda_n(y_2; w)$ is the output of the AINN reported in eq.(1). The Approximating Dynamical System is described in Fig. 9, where integrators, adders and multipliers have been realized by using analog circuits (Op. Amps., resistors and capacitors). This Approximating Dynamical System has been used to approximate the trajectories of the Duffing equation. This equation assumes a relevant importance since its behavior may be periodic, almost periodic or chaotic depending on the values of the parameters chosen.

Duffing equation is a simple form of nonlinear system which behave chaotically. These systems which were thoroughly studied in the past are described by the following equations:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\beta x_1^3 - \mu x_2 + A\cos\omega t \end{cases} \quad (3)$$

The Duffing Dynamical Systems has been implemented in analog hardware by using standard discrete components for values of the parameters β, μ and A for which its behaviour is periodic. Figs. 10 a-d reports the output of the Duffing dynamical system and the output of the approximating dynamical system during learning with CRWC.

4. Conclusions

A stochastic learning algorithm is presented in this work based on the idea that the Neural Network is partitioned in clusters and one cluster a time is trained. From the simulations performed we concluded that the partition guarantees an higher convergence probability.The algorithm has been used to train an analog neural network chip.

References:

[1] A.F.Murray, P.J.Edwards, "Enanched MLP Performance and Fault Tolerance Resulting from Synaptic Weigth Noise During Training", *IEEE Transactions on Neural Networks*, Vol.5,No.5, pp.792-802, September 1994.

[2] M.Conti, S.Orcioni, C.Turchetti, "A new stochastic learning algorithm for analog hardware implementation", *Proc. of the Int. Conf. on Artificial Neural Networks, ICANN 98*, Vol.2, pp.1171-1176.

[3]C.M.Bishop;Training with Noise is Equivalent to Tikhonov Regularization, *Neural Computation*, n.7, pp. 108-116, 1995.

[4] M.Conti, S.Orcioni, C.Turchetti, "A class of neural networks based on approximate identity for analog IC's hardware implementation", *IEICE Trans. on Fundamentals*,Vol.E77-A, n.6,pp.1069-1079, June 1994.

[5] M.Conti, "Analog CMOS implementation of Approximate Identity Neural Networks", *IEICE Trans. on Fundamentals of Electronics*, Feb 1997, pp.427-432.

[6] T.Chen, H.Chen, "Approximation of continuous functionals by neural networks with application to dynamic systems", *IEEE Trans. On Neural Networks*, Vol.4, No.6, Nov. 1993.

[7] M.Conti,C.Turchetti, "Approximation of dynamical systems by continuous-time recurrent approximate identity neural networks", *Int. Journal Neural, Parallel & Scientific Computations*, Vol.2, n.3, p.299,Sept. 1994.

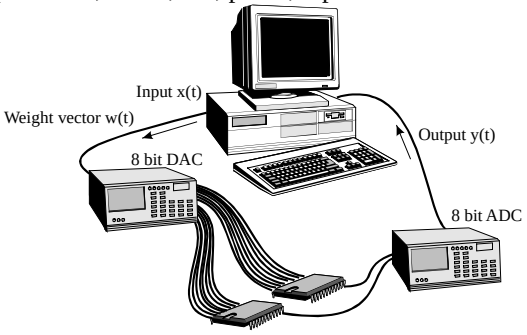
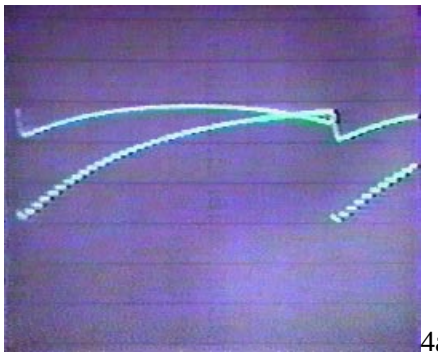
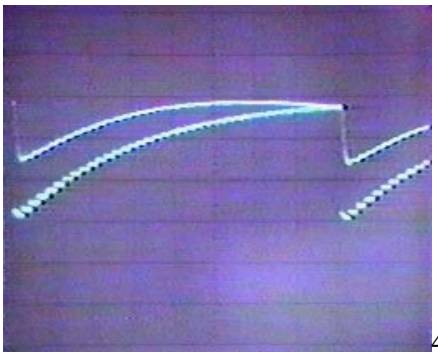


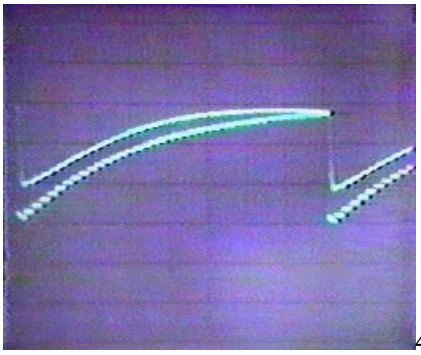
Fig. 3 Chip in the learning loop scheme.



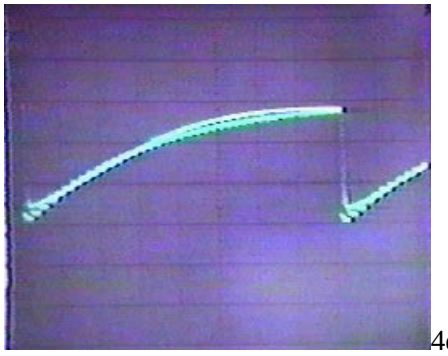
4a



4b

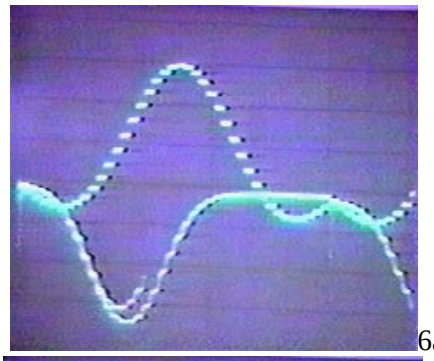


4c

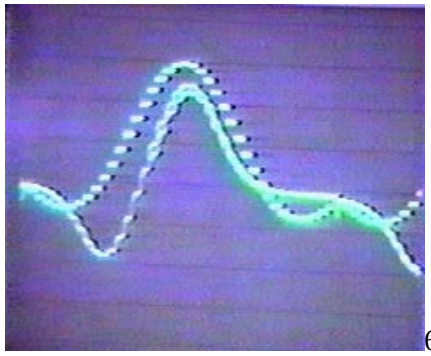


4d

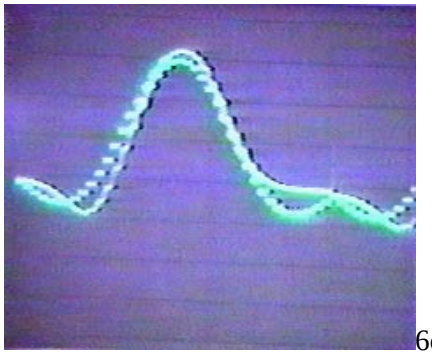
Fig. 4a-d. Output and desired output $y = 1-\exp(-x/\tau)$ during learning.



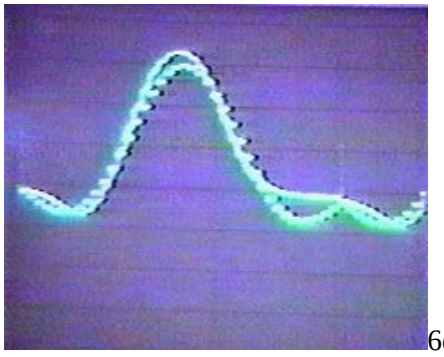
6a



6b



6c



6d

Fig. 6a-d. Output and desired output $y = \sin(\pi x)/x$ during learning.

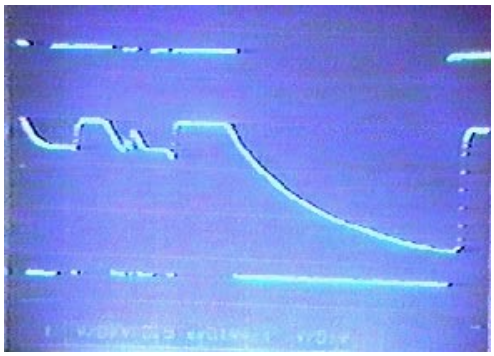


Fig. 5 Error during learning of the function of Figs 4.

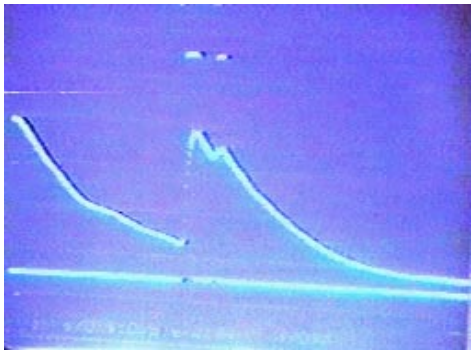
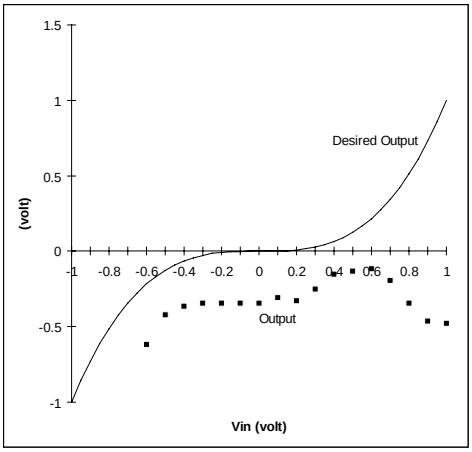
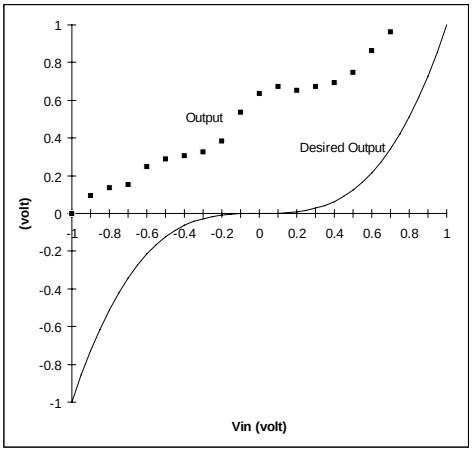


Fig. 7 Error during learning of the function of Figs 6.

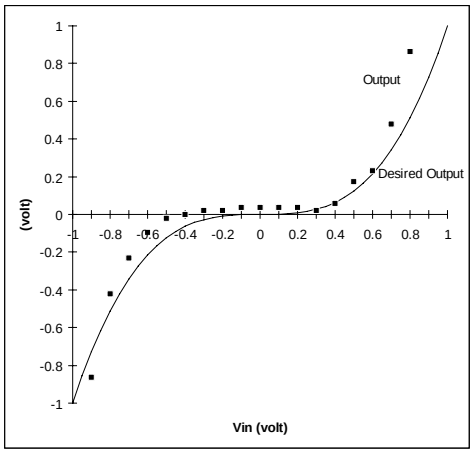
.



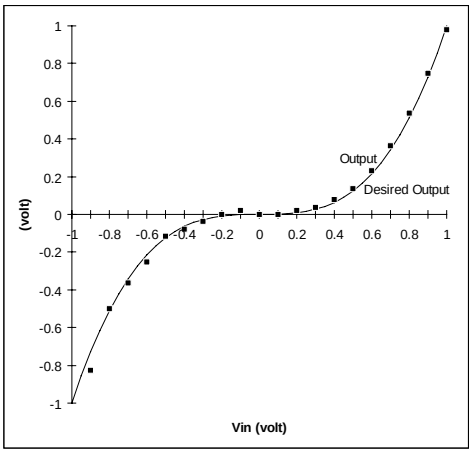
8a



8b



8c



8d

Fig. 8a-d. Output and desired output $y=x^3$ during learning.

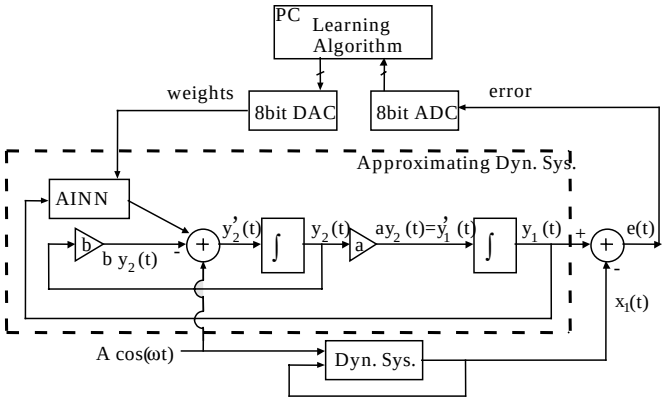
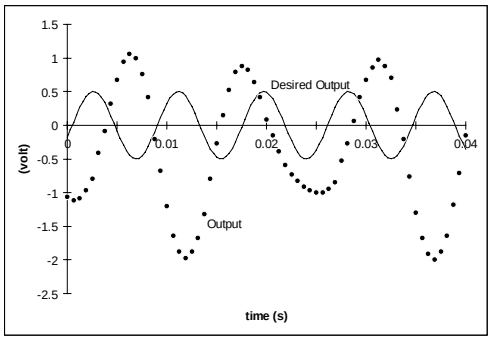
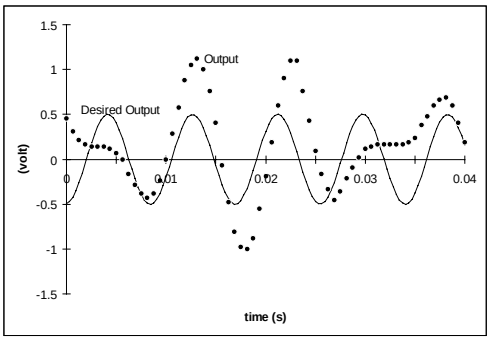


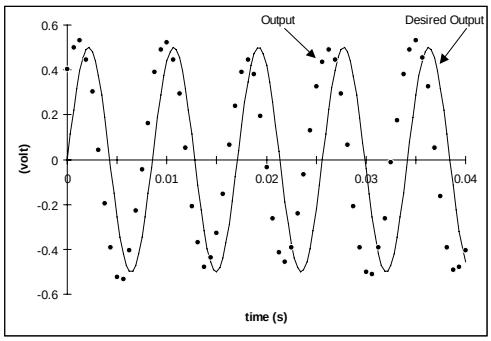
Fig. 9. Architecture used for the trajectory approximation.



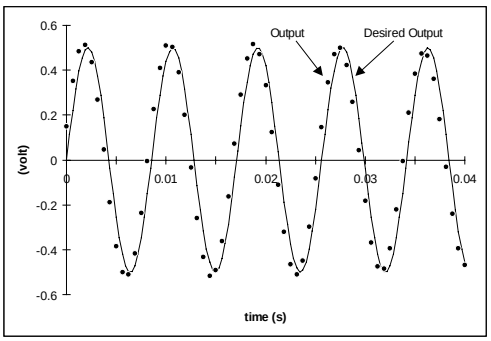
10a



10b



10c



10d

Fig. 10. Output and desired output trajectories during learning.