Control of Complex Nonlinear Dynamic Systems with Incremental Deep Learning Neural Networks

Antonio Moran
Pontifical Catholic University of Peru
amoran@pucp.edu.pe

Masao Nagai
Tokyo University of Agriculture and Technology
nagai@cc.tuat.ac.jp

***Abstract*—This paper proposes incremental deep learning methods for training neural networks to control the response of complex nonlinear dynamic systems. By analyzing the complexity of the task to be fulfilled by the neural network, learning strategies are formulated and implemented in an incremental scheme starting from simple tasks and continuing with increasingly complex tasks. The Dynamic Back Propagation algorithm is used for training the neuro-controller in each step of the incremental learning process, considering the system nonlinear dynamics. The results obtained in the control of highly unstable nonlinear systems, and the positioning control of mobile robots verify the effectiveness of the proposed incremental deep learning strategies.**

Keywords—recurrent neural networks, dynamic back propagation, neuro-control, incremental learning, mobile robots.

I. INTRODUCTION

Human beings are capable of performing complex tasks, acting autonomously for completing different activities in diverse environments and working conditions. Tasks like driving a car, diagnosing and curing a disease, writing and debugging a computer program can be completed by a person with the proper skills and experience.

For acquiring those skills, human beings have to complete an appropriate and structured training process starting from the easiest tasks towards the most difficult ones. When a person learns to drive a car, he/she starts driving in easy and simple conditions (no traffic, small car, etc.) to progressively continue with more difficult driving conditions (crowded roads, heavy traffic, multiple road signals, etc.). It is not conceivable to start driving training in complex environments or with difficult-to-drive cars. Thus, human beings acquire skills and expertise in an incremental and progressive way sequentially advancing from the easy and simple to the complex and difficult.

In the same way, when neural networks are to be trained to complete complex tasks, they have to be trained in a progressive scheme, starting with training for easy tasks and sequentially continuing with training for more complex tasks. Starting training a neural network directly for complex tasks usually does not yield expected results: learning does not converge or get trapped in undesirable local minima.

Several incremental learning strategies have been proposed for neural networks in supervised and unsupervised learning schemes depending on data size and availability, and complexity of the tasks to be performed. Depending on the particular characteristics of the problem, different incremental learning strategies can be formulated: decomposition into identifiable parts and components to be subsequently integrated, gradual availability of new data sets over time, identification of simple tasks that evolve into complex and more elaborated tasks, etc. In most cases, initially learned skills remain or improve as learning evolves in complexity and amount of data.

Incremental learning strategies have been applied for solving several pattern identification problems. An incremental learning strategy for face detection has been proposed in [1] where a convolutional neural network is progressively trained to detect parts of a human face to finally detect the complete integrated face. Incrementally building networks have been proposed in [2] and [3] where the number of layers and neurons of a neural network increase as new data becomes available in pattern classifications problems. Different incremental learning strategies are compared in [4] for training convolutional neural networks where data sets are available in consecutive batches, and retraining the model from scratch is unfeasible.

Also, incremental deep learning has been applied to solve complex dynamic problems in robotics, automation and control engineering. Incremental learning methods are proposed in [5], [6] and [7] for the accurate dynamic modeling and real-time control of robot manipulators. Incremental learning of Takagi–Sugeno fuzzy-neural networks have been proposed in [8] and [9] for the creation and updating of membership functions and fuzzy rules. Beginning with simple Takagi-Sugeno fuzzy-neural networks, the algorithm develops the network structure by adding more fuzzy terms and rules for incrementally improving the control performance.

This paper proposes and analyzes several incremental deep learning strategies for training neural networks for solving different problems in control, automation and robotics. Neural networks are sequentially trained starting from simple tasks towards complex tasks using the Dynamic Back Propagation DBP algorithm, taking into account the dynamics of the system to be controlled [10], [11]. The following control problems are analyzed and solved through incremental deep learning: (1) stabilization and tracking control of a weakly unstable nonlinear system, (2) stabilization of a highly unstable nonlinear system, (3) positioning and path following control of a car-like mobile robot. For each problem an incremental learning strategy is proposed which is composed by several learning steps each

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

one defined so that learning is feasible. Initial learning steps are easier to complete, and the trained neural network is used as the starting point for the next learning step, and so on. Obtained results show that the proposed incremental deep learning strategies are effective for training neural networks with the Dynamic Back Propagation algorithm but the formulation of strategies depends on the particular characteristics of the problem and of the dynamic system to be controlled.

II. DYNAMIC BACK PROPAGATION BASED CONTROL OF NONLINEAR SYSTEMS

A. Control Problem

The objective of controlling a dynamic system is to make it behaves in a desired way. For that, a controller is designed for providing the inputs to the system so that it responds as desired. In this paper, a neural network will be trained to perform as the controller of a dynamic system as its shown in Figure 1 where a neuro-controller, having as inputs the system state $\mathbf{x}$ and desired state $\mathbf{x}^*$, provides the control input u for properly driving the system towards the desired state to complete a given task. The neuro-controller implements a control law determined considering the dynamics of the system to be controlled, as well as the dynamics of the closed-loop feedback control system.

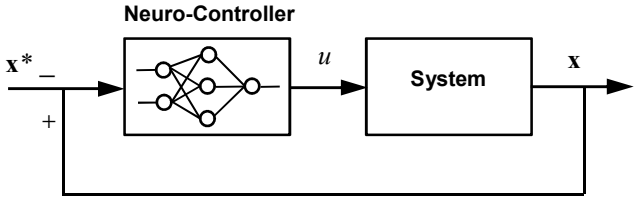

Figure. 1. General structure of a feedback neuro-control system.

Given a nonlinear system described by the following discrete-time state equation:

$$\mathbf{x}_{k+1} = \boldsymbol{f}(\mathbf{x}_k, u_k) \tag{1}$$

where the state $\mathbf{x} \in R^n$, the control input $u \in R$, and the vector function $\boldsymbol{f}(\mathbf{x}_k, u_k)$ has continuous derivatives, the control objective is to determine a state-dependent control law capable to drive the system from any initial state to the final desired state $\mathbf{x}^* = 0$ (stabilization problem). The feedback control law is given by the following expression to be represented by a neural network (neuro-controller):

$$u_k = k(\mathbf{x}_k) \tag{2}$$

where $k(\mathbf{x}_k)$ is a function $R^n \rightarrow R$. If the control objective is to drive the system to the desired state $\mathbf{x}^* \neq 0$ (state $\mathbf{x}$ converge to $\mathbf{x}^*$), the control law of (2) is modified to be:

$$u_k - u^* = k(\mathbf{x}_k - \mathbf{x}^*) \tag{3}$$

where u^* is the convergence value of the control input. Since $\mathbf{x}^*$ and u^* are convergence values, they have to be chosen to satisfy the following equation, from (1):

$$\mathbf{x}^* = \boldsymbol{f}(\mathbf{x}^*, u^*) \tag{4}$$

It is clear that when $\mathbf{x}^* = \mathbf{0}$ and $u^* = 0$, equation (4) turns to be:

$$\mathbf{0} = \boldsymbol{f}(\mathbf{0}, 0) \tag{5}$$

which is a condition to be fulfilled by the system model (1).

Controller $u_k = k(\mathbf{x}_k)$ in (2) is determined by minimizing a cost function defined in terms of control objectives. In general, the cost function can be defined by the following quadratic form:

$$J = \frac{1}{2}\sum_{k=0}^{N} (\mathbf{x}_k - \mathbf{x}_k^*)^T \mathbf{Q} (\mathbf{x}_k - \mathbf{x}_k^*) + u_k^T \mathrm{R} u_k \tag{6}$$

where $\mathbf{x}_k^*$ represents the desired state at step k, and $\mathbf{Q}$ and R are square, symmetric and positive matrices of appropriate dimensions. Index N is chosen large enough to ensure the desired state $\mathbf{x}_k^*$ is attained.

From optimal control theory it is known that matrix R must have inverse in order to have bounded values of control input u_k. In order to simplify the neuro-controller training, it will be considered R small, and training will be continuously monitored for ensuring bounded values of u_k and convergence of the learning process.

B. Dynamic Training of Neuro-Controller

The neuro-controller will be trained using the Dynamic Back Propagation algorithm considering the dynamics of the system to be controlled, as well as the dynamics of the closed-loop feedback control system as it is shown in Figure 2, in a similar structure as in [10] and [11]. The neuro-controller is represented by a multilayer network with one hidden layer and weighting coefficients represented by v and w to be determined. Neurons in input and output layers are linear whereas hidden layer neurons are nonlinear.

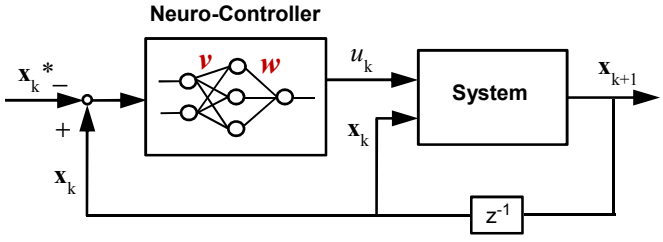

Figure 2. Feedback control system highlighting the involved dynamics.

The neuro-controller will be trained for minimizing the cost function J given in (6), and the weightings coefficients v and w of the neuro-controller will be updated using the gradient descend algorithm:

$$v = v - \eta \frac{\overline{\partial J}}{\partial v} \quad , \quad w = w - \eta \frac{\overline{\partial J}}{\partial w} \tag{7}$$

where η is the learning rate, and $\frac{\overline{\partial J}}{\partial v}$ and $\frac{\overline{\partial J}}{\partial w}$ are **total** partial derivatives that take into account the dynamics of the feedback control problem. A clear definition of total and simple partial derivatives can be found in [10]. From (6) and considering R small, the total partial derivatives $\frac{\overline{\partial J}}{\partial v}$ and $\frac{\overline{\partial J}}{\partial w}$ can be computed using the following expressions:

$$\frac{\overline{\partial J}}{\partial v} = \sum_{k=0}^{N} (\mathbf{x}_k - \mathbf{x}_k^*)^T \mathbf{Q} \frac{\overline{\partial \mathbf{x}_k}}{\partial v} \tag{8}$$

$$\frac{\overline{\partial J}}{\partial w} = \sum_{k=0}^{N} (\mathbf{x}_k - \mathbf{x}_k^*)^T \mathbf{Q} \frac{\overline{\partial \mathbf{x}_k}}{\partial w} \tag{9}$$

where $\frac{\overline{\partial \mathbf{x}_k}}{\partial v}$ and $\frac{\overline{\partial \mathbf{x}_k}}{\partial w}$ are the total partial derivatives of $\mathbf{x}_k$ respect to neuro-controller weightings v and w, respectively.

The recursive equations for updating de values of $\frac{\overline{\partial \mathbf{x}_k}}{\partial v}$ and $\frac{\overline{\partial \mathbf{x}_k}}{\partial w}$ are given by the following expressions [10]:

$$\frac{\overline{\partial \mathbf{x}_{k+1}}}{\partial v} = \frac{\partial \mathbf{x}_{k+1}}{\partial u_k}\frac{\partial u_k}{\partial v} + \left(\frac{\partial \mathbf{x}_{k+1}}{\partial \mathbf{x}_k} + \frac{\partial \mathbf{x}_{k+1}}{\partial u_k}\frac{\partial u_k}{\partial \mathbf{x}_k}\right)\frac{\overline{\partial \mathbf{x}_k}}{\partial v} \tag{10}$$

$$\frac{\overline{\partial \mathbf{x}_{k+1}}}{\partial w} = \frac{\partial \mathbf{x}_{k+1}}{\partial u_k}\frac{\partial u_k}{\partial w} + \left(\frac{\partial \mathbf{x}_{k+1}}{\partial \mathbf{x}_k} + \frac{\partial \mathbf{x}_{k+1}}{\partial u_k}\frac{\partial u_k}{\partial \mathbf{x}_k}\right)\frac{\overline{\partial \mathbf{x}_k}}{\partial w} \tag{11}$$

Derivatives $\frac{\partial \mathbf{x}_{k+1}}{\partial u_k}$ and $\frac{\partial \mathbf{x}_{k+1}}{\partial \mathbf{x}_k}$ are computed from the system state equation given by (1), and derivatives $\frac{\partial u_k}{\partial w}$, $\frac{\partial u_k}{\partial v}$ and $\frac{\partial u_k}{\partial \mathbf{x}_k}$ are computed from the neuro-controller.

Initial values of $\frac{\overline{\partial \mathbf{x}_k}}{\partial v}$ and $\frac{\overline{\partial \mathbf{x}_k}}{\partial w}$ are set to zero.

For a set of initial state variables, and initial random weighting coefficients v and w, the system runs from $k = 0$ to $k = N$ for computing the state $\mathbf{x}_k$ and control input u_k which are used to compute the total partial derivatives $\frac{\overline{\partial \mathbf{x}_k}}{\partial v}$ and $\frac{\overline{\partial \mathbf{x}_k}}{\partial w}$ of (10) and (11), which are afterwards used for computing the total partial derivatives $\frac{\overline{\partial J}}{\partial v}$ and $\frac{\overline{\partial J}}{\partial w}$ of (8) and (9) required for updating the weightings coefficients v and w in (7). This process is repeated consecutively until the neuro-controller is finally trained for properly controlling the dynamic system.

III. Problem 1: Stabilization and Tracking Control of a Weakly Unstable Nonlinear System

Given the weakly unstable bilinear system:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}u_k + \mathbf{G}\mathbf{x}_k u_k \tag{12}$$

with matrices **A**, **B** and **G** given by:

$$\mathbf{A} = \begin{bmatrix} 1.01 & 0.15 \\ -0.25 & 1.00 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0.0 \\ 1.5 \end{bmatrix} \quad \mathbf{G} = \begin{bmatrix} 0.3 & 0.1 \\ -0.1 & 0.2 \end{bmatrix} \tag{13}$$

The eigenvalues of matrix **A** are equal to $1.0050 \pm 0.1936j$ with module slightly greater than 1 (slightly outside the unit circle in the complex plane). From systems theory, it is known that a linear system with the matrix **A** of (13) is unstable. It can also be shown that the bilinear system (12) with matrices **A**, **B** and **G** is also unstable [12].

The control objective is to determine a feedback control law (2) to stabilize the system from any initial state in the range:

$$-3 \le x_{1(0)} \le 3$$
$$-3 \le x_{2(0)} \le 3 \tag{14}$$

Training a neuro-controller for this weakly unstable system is not an easy task because learning is not assured to converge given the broad range of initial values of state variables x_1 and x_2 as shown in (14), and considering the fact that initial values of neuro-controller weightings v and w are randomly selected. Training convergence can be made feasible if incremental learning is applied. The question is: how should incremental learning be implemented?

Since the control objective is to drive the system from an arbitrary initial state (inside the range given in (14)) to the desired final states $x_1^* = 0$ and $x_2^* = 0$ (stabilization problem), it is expected that learning is more probable to converge if the initial states are close (or very close) to the desired states. In other words, the farther the initial state variables are from the desired states, the more difficult learning convergence will be. Considering this characteristic of the problem, the incremental learning strategy is proposed as shown in Table 1:

Table 1. Incremental learning strategy respect to initial states $x_{1(0)}$ and $x_{2(0)}$

1. Fix initial states close to the desired states $x_1^* = 0$ and $x_2^* = 0$.
2. Train the neuro-controller using the Dynamic Back Propagation algorithm.
3. If learning converges, increase the value of initial states and go to step 2. If the upper and lower limits of $x_{1(0)}$ and $x_{2(0)}$ given by (14) are achieved, then learning is complete.
4. If learning does not converge, reduce the value of initial states and go to step 2.

In incremental learning respect to the initial states $x_{1(0)}$ and $x_{2(0)}$, the values shown in Table 2 were considered along the different steps of the learning process: the first two columns correspond to the values of initial states $x_{1(0)}$ and $x_{2(0)}$ at the beginning of the process, and the last two

columns correspond to the final upper and lower bounds of initial states $x_{1(0)}$ and $x_{2(0)}$ given in (14). Table 2 shows the four sets of initial state variables considered at each step of the incremental learning process.

Table 2. Variation of initial states $x_{1(0)}$ and $x_{2(0)}$ along the incremental learning process.

$x_{1(0)}$	$x_{2(0)}$		$x_{1(0)}$	$x_{2(0)}$		$x_{1(0)}$	$x_{2(0)}$
...	...		...	...		...	...
−0.1	−0.1	→	−0.5	−0.5	→ ... →	−3.0	−3.0
−0.1	0.1		−0.5	0.5		−3.0	3.0
0.1	−0.1		0.5	−0.5		3.0	−3.0
0.1	0.1		0.5	0.5		3.0	3.0

The size of change of initial states $x_{1(0)}$ and $x_{2(0)}$ in Table 2 should be small enough for ensuring learning to converge. If convergence is not achieved, change size is reduced.

Derivatives $\frac{\partial \mathbf{x}_{k+1}}{\partial \mathbf{x}_k}$ and $\frac{\partial \mathbf{x}_{k+1}}{\partial u_k}$ required in recursive expressions (10) and (11) can be computed as follows:

$$\frac{\partial \mathbf{x}_{k+1}}{\partial \mathbf{x}_k} = \mathbf{A} + \mathbf{G}u_k$$
$$\frac{\partial \mathbf{x}_{k+1}}{\partial u_k} = \mathbf{B} + \mathbf{G}\mathbf{x}_k \quad (15)$$

Figure 3 shows the time response of state variables x_1 and x_2 for two different initial conditions: (a) $x_{1(0)} = 3, x_{2(0)} = 3$ and (b) $x_{1(0)} = -3$, $x_{2(0)} = 3$. These initial conditions correspond to the borders of the range given in (14). It is noted that the neuro-controller is able to stabilize the system whose states asymptotically converge to zero. Figure 3(a) and 3(b) also show the time response of the control input u which is smooth and bounded for all responses. These results verify the effectiveness of the proposed incremental learning strategy and the Dynamic Back Propagation algorithm.

The same neuro-controller (trained for stabilization, i.e. $x_1^* = 0$ and $x_2^* = 0$) can also be used if the desired states are different to zero (tracking problem, i.e. $x_1^* \neq 0$ and $x_2^* \neq 0$). As it was explained in Section II, the control law is modified to be $u_k - u^* = k(\mathbf{x}_k - \mathbf{x}^*)$ where $\mathbf{x}^*$ and u^* are convergence values chosen to satisfy the equation $\mathbf{x}^* = \boldsymbol{f}(\mathbf{x}^*, u^*)$.

For the particular desired convergence state $x_1^* = 1$, the convergence values of x_2^* and u^* are computed to be: $x_2^* = -0.3932$ and $u^* = 0.1892$. The time response is shown in Figure 4 where it is noted that variable x_1 converges to the desired state $x_1^* = 1$, and variable x_2 and control input u converge to x_2^* and u^*, respectively, when the system starts from initial conditions equal to zero. These results also verify the effectiveness of the proposed incremental learning strategy for training the neuro-controller.

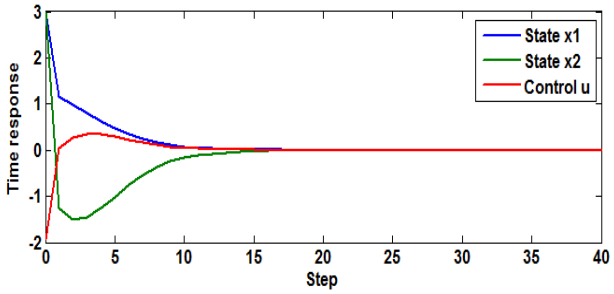

Figure 3(a). Time response of x_1, x_2 and u for initial condition: $x_{1(0)} = 3$, $x_{2(0)} = 3$.

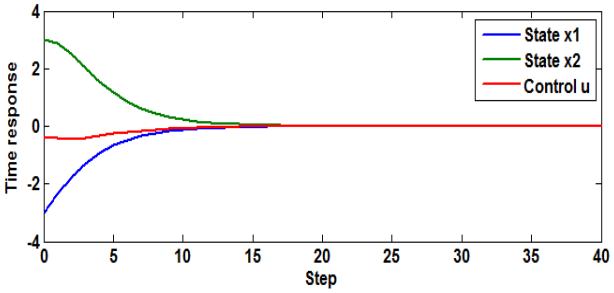

Figure 3(b). Time response of x_1, x_2 and u for initial condition $x_{1(0)} = -3$, $x_{2(0)} = 3$.

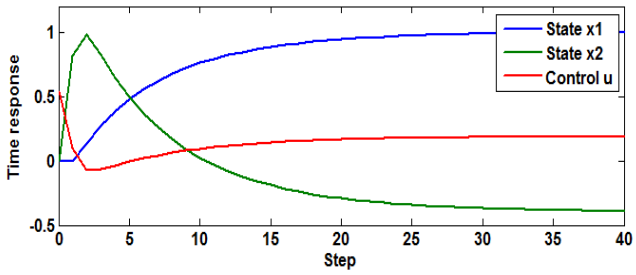

Figure 4. Time response of x_1, x_2 and u for initial condition $x_{1(0)} = 0$, $x_{2(0)} = 0$ and desired final state $x_1^* = 1$.

IV. Problem 2: Stabilization of a Highly Unstable Nonlinear System

Given the highly unstable bilinear system:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}u_k + \mathbf{G}\mathbf{x}_k u_k \quad (16)$$

with matrices **A**, **B** and **G** given by:

$$\mathbf{A} = \begin{bmatrix} 1.30 & 0.15 \\ -0.25 & 1.30 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0.0 \\ 1.5 \end{bmatrix} \quad \mathbf{G} = \begin{bmatrix} 0.3 & 0.1 \\ -0.1 & 0.2 \end{bmatrix} \quad (17)$$

The eigenvalues of matrix **A** are equal to $1.3000 \pm 0.1936j$ with module equal to 1.3143 greater than 1 (outside the unit circle in the complex plane). The problem is similar to that analyzed in Section III, being the only difference matrix **A** which has eigenvalues farther from the unit circle, and, therefore, the system is more unstable [12].

The control objective is to determine a feedback control law (2) to stabilize the system from any initial state in the range:

$$-3 \leq x_{1(0)} \leq 3$$

$$-3 \leq x_{2(0)} \leq 3 \quad (18)$$

Training a neuro-controller for this highly unstable nonlinear system is not an easy task because learning is not assured to converge given the broad range of initial values of state variables x_1 and x_2 as shown in (18), and considering the fact that the initial values of neuro-controller weighting coefficients v and w are randomly selected. Training convergence can be made feasible if incremental learning is applied. The question is how should incremental learning be implemented?

Two points are of relevance in this analysis:

- Highly unstable matrix **A**, equation (17)
- Broad range of initial values of state variables (18)

The more unstable matrix **A** is the more difficult learning is to converge. The farther the initial state variables are from the desired states (zero states for stabilization problem), the more difficult learning is to converge.

Considering these characteristics of the problem, the incremental learning strategy will have two components: incremental learning respect to matrix **A** (from not too unstable -or stable- matrix **A** towards the original highly unstable matrix **A**), and incremental learning respect to the initial values of state variables (from initial state variables close to zero towards the bounds of the range of the state variables). The last component is similar to the problem analyzed in Section III. Considering these characteristics of the problem, the incremental learning strategy is proposed in Table 3.

Table 3. Incremental learning strategy respect to matrix **A** and respect to initial states $x_{1(0)}$ and $x_{2(0)}$

1. Fix initial states $x_{1(0)}$ and $x_{2(0)}$ close to the desired states (close to zero given that it is a stabilization problem).
2. Adjust coefficients of matrix **A** to make it not too unstable (or stable).
3. Train neural network using the Dynamic Back Propagation algorithm.
4. If learning converges, adjust matrix **A** closer to its original value, then go to step 3. If matrix **A** has recovered its original value, then go to step 6.
5. If learning does not converge, go to step 2, adjusting matrix **A** to make it less unstable (or stable) until learning converges.
6. With the original value of matrix **A** recovered, increase the values of the initial states $x_{1(0)}$ and $x_{2(0)}$.
7. Train the neural network using the Dynamic Back Propagation algorithm.
8. If learning converges, increase the value of initial states $x_{1(0)}$ and $x_{2(0)}$, then go to step 7. If the upper and lower bounds of $x_{1(0)}$ and $x_{2(0)}$ are achieved, then learning is complete.
9. If learning does not converge, reduces the values of the initial states and go to step 7.

In incremental learning respect to matrix **A,** the matrices shown in Table 4 were considered along the different steps of the learning process. It is noted that the first matrix is chosen to be weakly unstable (or stable) and the last matrix corresponds to the original highly unstable matrix **A**.

Table 4. Variation of matrix **A** along incremental learning process.

$$\mathbf{A} = \begin{bmatrix} 1.01 & 0.15 \\ -0.25 & 1.01 \end{bmatrix} \rightarrow \begin{bmatrix} 1.05 & 0.15 \\ -0.25 & 1.05 \end{bmatrix} \cdots \rightarrow \begin{bmatrix} 1.30 & 0.15 \\ -0.25 & 1.30 \end{bmatrix}$$

The size of change of matrix **A** along the incremental learning process (Table 4) should be small enough for ensuring learning to converge. If convergence is not achieved, change size is reduced.

In incremental learning respect to the initial states $x_{1(0)}$ and $x_{2(0)}$, the values shown in Table 5 were considered along the different steps of the learning process. It is noted that the last two columns correspond to the upper and lower bounds of the initial states of $x_{1(0)}$ and $x_{2(0)}$ given in (18).

Table 5. Variation of initial states $x_{1(0)}$ and $x_{2(0)}$ along the incremental learning process.

$x_{1(0)}$	$x_{2(0)}$		$x_{1(0)}$	$x_{2(0)}$		$x_{1(0)}$	$x_{2(0)}$
...	...		...	...		...	...
−0.1	−0.1		−0.5	−0.5		−3.0	−3.0
−0.1	0.1	→	−0.5	0.5	→ ... →	−3.0	3.0
0.1	−0.1		0.5	−0.5		3.0	−3.0
0.1	0.1		0.5	0.5		3.0	3.0

The size of the change of initial states $x_{1(0)}$ and $x_{2(0)}$ along the incremental learning process should be small enough for ensuring learning to converge. If convergence is not achieved, change size is reduced.

Figure 5 shows the time response of state variables x_1 and x_2 for two different initial conditions: (a) $x_{1(0)} = 3, x_{2(0)} = 3$, and (b) $x_{1(0)} = -3, x_{2(0)} = 3$. These initial conditions belong to the borders of the range given by (18). It is noted that the neuro-controller is able to stabilize the system whose states asymptotically converge to zero. Figure 5 also shows the time response of the control input u which is smooth and bounded. These results verify the effectiveness of the two-steps incremental learning strategy and the Dynamic Back Propagation algorithm.

V. PROBLEM 3: AUTONOMOUS POSITIONING OF A MOBILE ROBOT

Given a car-like mobile robot, the objective is to train a neuro-controller for driving the robot from any initial position to a final desired position, as it is shown in Figure 6. The variables defining the position of the mobile robot are shown in Figure 7: coordinates (x, y) of the rear center of the car, and inclination angle θ measured respect to the horizontal X-axis.

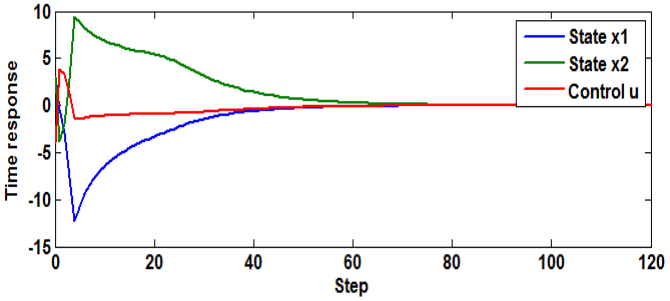

Figure 5(a). Time response of x_1 , x_2 and u for the initial condition $x_{1(0)} = 3$, $x_{2(0)} = 3$.

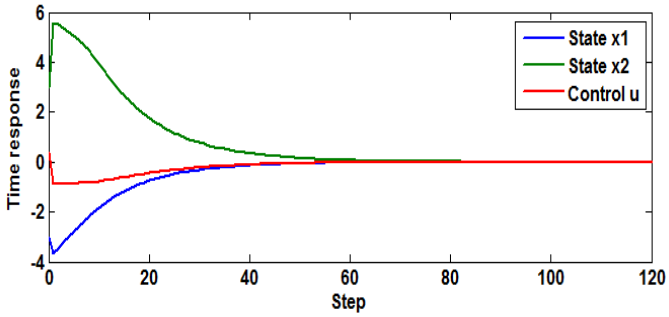

Figure 5(b). Time response of x_1 , x_2 and u for the initial condition $x_{1(0)} = -3$, $x_{2(0)} = 3$.

The steering angle of front wheels δ is the input to the mobile robot.

Considering backward motion at constant low speed, rear traction, front steering, and no-slipping/no-skidding, the discrete-time equations describing the motion of the mobile robot are [13], [14]:

$$x_{k+1} = x_k + r\cos\theta \tag{19}$$

$$y_{k+1} = y_k + r\sin\theta \tag{20}$$

$$\theta_{k+1} = \theta_k - (r/L)\tan\delta \tag{21}$$

where r is the distance the rear wheels move at each step, and L is the length of the mobile robot.

The control objective is to train a neural network (neuro-controller) to act as a driver providing the steering angle input δ for conducting the mobile robot from any initial position to the desired final position. The initial position is given by state variables x_0, y_0, θ_0, and the final position is given by state variables $x^* = 0, y^* = 30, \theta^* = \pi/2$ as it is shown in Figure 6. The neuro-controller should be trained so that the mobile robot starts from any initial position with coordinate x_0 in the range:

$$-20 \le x_0 \le 20 \tag{22}$$

The steering angle δ must be in the range:

$$-\pi/4 \le \delta \le \pi/4 \tag{23}$$

Also, considering the desired inclination angle $\theta^* = \pi/2$, the range of θ is chosen to be:

$$-\pi/2 \le \theta \le 3\pi/2 \quad \text{(full turn)} \tag{24}$$

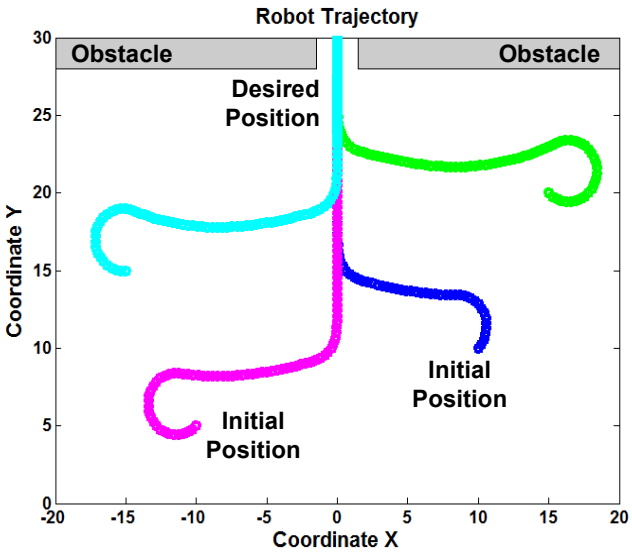

Figure 6. Car-like mobile robot positioning problem. Desired position: $x^* = 0, y^* = 30, \theta^* = \pi/2$

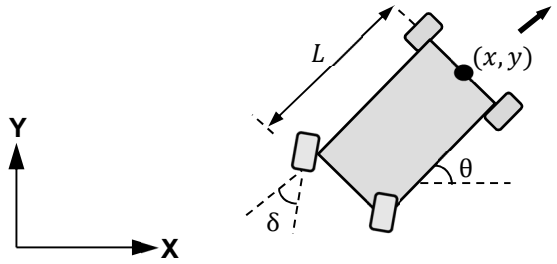

Figure 7. Variables of car-like mobile robot.

Considering the characteristics of the car-like robot motion, the problem can be solved using only coordinate x and inclination angle θ for computing the required steering angle δ at every step of motion, as it has been applied in [13], [14], [15]. In this way coordinate y is not required as input to the neuro-controller. This consideration is applicable as far as there is enough distance between the initial and desired positions.

Training a neuro-controller to perform well for the complete range of initial coordinate x_0 as in (22) is not an easy task because learning is not assured to converge given that the initial values of neuro-controller weighting coefficients v and w are randomly selected. Training convergence can be made feasible if incremental learning is applied. The question is how should incremental learning be implemented?

Similarly as the analysis of Section III, it is clear that the farther the initial coordinate x_0 is from the desired coordinate $x^* = 0$, the more difficult the learning is to converge. Considering this characteristic of the problem, the incremental learning strategy is proposed as follows in Table 6:

Table 6. Incremental learning strategy respect to initial coordinate $x_{(0)}$.

1. Fix initial states for training close to the desired states $x^* = 0$ and $\theta^* = \pi/2$.
2. Train neuro-controller using the Dynamic Back Propagation algorithm.
3. If learning converges, increase the value of initial coordinate x_0 and go to step 2. If the upper and lower bounds of x_0 are achieved, then learning is complete.
4. If learning does not converge, reduce the value of initial coordinate x_0 and go to step 2.

In incremental learning respect to the initial coordinate x_0, and θ_0, the values shown in Table 7 were considered along the different steps of the learning process. It is noted that the last column of coordinate x_0 corresponds to the upper and lower bounds given in (22). It is also noted that the initial value of θ_0 remains the same in all learning steps.

Table 7. Variation of initial coordinate $x_{(0)}$ along incremental learning process.

x_0	θ_0		x_0	θ_0			x_0	θ_0
...	...		...	...			...	...
-2	$-\pi/2$	$\rightarrow$	-4	$-\pi/2$	$\rightarrow$	... $\rightarrow$	-20	$-\pi/2$
-2	$\pi/2$		-4	$\pi/2$			-20	$\pi/2$
2	$-\pi/2$		4	$-\pi/2$			20	$-\pi/2$
2	$\pi/2$		4	$\pi/2$			20	$\pi/2$

The size of the change of initial coordinate x_0 in Table 7 should be small enough for ensuring the learning to converge. If convergence is not achieved, change size is reduced.

For applying the Dynamic Back Propagation algorithm, the following state space equation is used (from (19) and (21)), without considering coordinate y_k:

$$\begin{bmatrix} x_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + r\cos\theta \\ \theta_k - \frac{r}{L}u_k \end{bmatrix}$$

where $u_k = \tan\delta_k$

Considering the state vector $\mathbf{x} = [x \quad \theta]^T$ derivatives $\frac{\partial \mathbf{x}_{k+1}}{\partial \mathbf{x}_k}$ and $\frac{\partial \mathbf{x}_{k+1}}{\partial u_k}$ required in (10) and (11) are computed as follows:

$$\frac{\partial \mathbf{x}_{k+1}}{\partial \mathbf{x}_k} = \begin{bmatrix} 1 & -r\sin\theta \\ 0 & 1 \end{bmatrix} \tag{25}$$

$$\frac{\partial \mathbf{x}_{k+1}}{\partial u_k} = \begin{bmatrix} 0 \\ -\frac{r}{L} \end{bmatrix} \tag{26}$$

After incremental learning was completed, the performance of the neuro-controller was verified for different initial positions as it is shown in Figure 6 where it is noted that the mobile robot smoothly moves towards the final desired position and reach it without colliding with obstacles at both sides of the desired position.

The neuro-controller can also be used to drive the mobile robot to other final positions as it is shown in Figure 8 where two different desired positions are considered: (a) $x^* = -10$, $y^* = 30$, $\theta^* = \pi/2$ and (b) $x^* = 5$, $y^* = 30$, $\theta^* = \pi/2$. It is noted the neuro-controller is able to conduct the mobile robot to the desired positions without colliding with the obstacles at both sides.

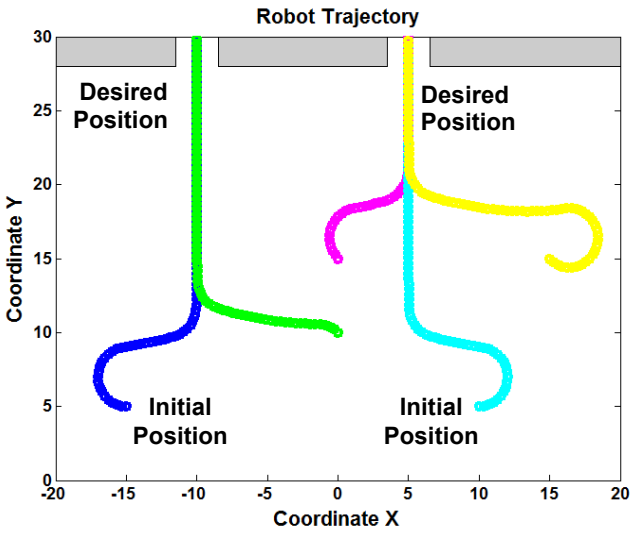

Figure 8. Trajectories of mobile robot for two desired positions: (a) $x^* = -10$, $y^* = 30$, $\theta^* = \pi/2$, and (b) $x^* = 5$, $y^* = 30$, $\theta^* = \pi/2$

Also, the same neuro-controller can be used for path following as it is shown in Figure 9 where the mobile robot follows a circular path with center in coordinates (20,0) and radius $R = 30$. For this, it is only required to change the values of the desired coordinate x^* and desired inclination θ^* at each motion step, as it is stated in equation (3). Also, for circular motion, the steering angle converges to $\delta^* \neq 0$. As it is presented in [15], the values of x^*, θ^* and δ^* are given by:

$$x^* = 20 - \frac{R(20-x)}{\sqrt{(20-x)^2 + y^2}} \tag{27}$$

$$\theta^* = \operatorname{atan}\left(\frac{20-x}{y}\right) \tag{28}$$

$$\delta^* = \operatorname{atan}\left(\frac{L}{R}\right) \tag{29}$$

Figure 9 shows the trajectory of the mobile robot following a circular path, starting from two different initial positions: (a) $x_0 = -15$, $y_0 = 20$, $\theta_0 = 5\pi/4$ and (b) $x_0 = 10$, $y_0 = 5$, $\theta_0 = \pi/4$. It is noted the neuro-controller is able to conduct the mobile robot for following the desired circular trajectory.

Figure 10 shows the time response of the steering angle δ for the trajectory corresponding to initial position (b). It is noted the steering angle is bounded between -45 and $+45$ degrees. These results verify the effectiveness of the incremental learning strategy and the Dynamic Back Propagation algorithm.

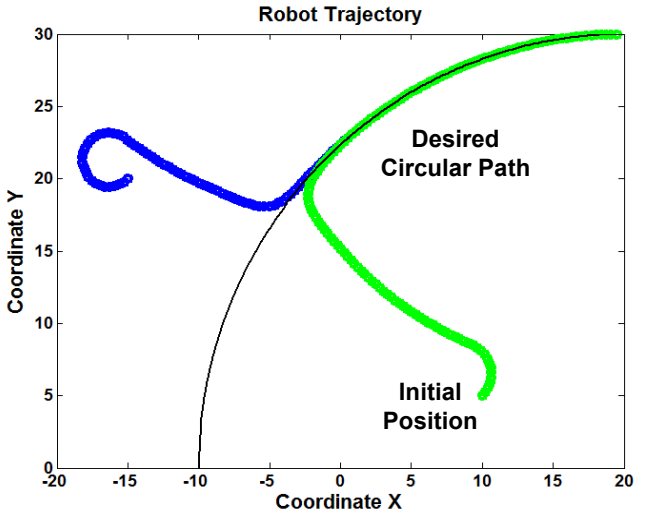

Figure 9. Trajectories of mobile robot for circular path following. Two different initial positions: (a) $x_0 = -15$, $y_0 = 20$, $\theta_0 = 5\pi/4$, and (b) $x_0 = 10$, $y_0 = 5$, $\theta_0 = \pi/4$.

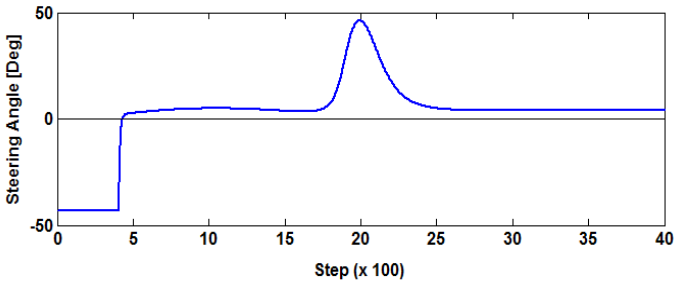

Figure 10. Time response of steering angle δ for circular path following, starting from initial position $x_0 = 10$, $y_0 = 5$, $\theta_0 = \pi/4$.

VI. CONCLUSIONS

The problem of training neural networks for controlling nonlinear dynamic systems has been analyzed. It was found that learning for complex tasks can be made possible if it is implemented in an incremental scheme, starting from easy and simple tasks and sequentially continuing with increasingly complex and difficult tasks. The strategy for incremental learning depends on the particular characteristics of the problem understanding what is easy and simple for the system to be controlled.

For training the neuro-controller the Dynamic Back Propagation algorithm was used in each step of the incremental learning process. The good results obtained for the control of highly unstable nonlinear systems, and the positioning control of a mobile robot verify the effectiveness of the proposed incremental learning strategies.

REFERENCES

[1] D. Triantafyllidou and A. Tefas, "Face detection based on deep convolutional neural networks exploiting incremental facial part learning", 23rd International Conference on Pattern Recognition ICPR 2016, Mexico, December 2016.

[2] S. Bose and M. Huber, "Incremental learning of neural network classifier using reinforcement learning", IEEE International Conference on Systems, Man, and Cybernetics, Budapest, October, 2016.

[3] Y. Wong, K, Seng, and L. Ang, "Radial basis function neural network with incremental learning for face recognition", IEEE Transaction on Systems, Man, and Cybernetics, Vol.41, No.4, August 2011.

[4] V. Lomonaco and D. Maltoni, "Comparing incremental learning strategies for convolutional neural networks", Artificial Neural Networks in Pattern Recognition ANPR 2016, Lecture Notes in Computer Sciences, Springer.

[5] D. Lee, "Incremental robot skill learning by human retargetting and physical human guidance", 12th International Conference on Ubiquitous Robots and Ambient Intelligence URAI 2015, Goyang, South Korea, October 2015.

[6] Z. Wang, C. Chen, H. Li, D. Dong, T. Tarn, "A novel incremental learning scheme for reinforcement learning in dynamic environments", 12th World Congress on Intelligent Control and Automation WCICA 2016, Guilin, China, 2016.

[7] "Incremental learning of robot dynamics using random features", IEEE International Conference on Robotics and Automation ICRA 2016, Shanghai, China, May 2011.

[8] E. Lughofer, "FLEXFLIS: A robot incremental learning approach for evolving Takagi-Sugeno fuzzy models", IEEE Transaction on Fuzzy Systems, Vol.16, No.6, 2008.

[9] D. Wang, X. Zeng, J. Keane, "An incremental construction learning algorithm for identification of T-S fuzzy systems", IEEE International Conference Fuzzy Systems FUZZY-IEEE 2018, Hong Kong, China, 2018.

[10] K. Narendra, and K. Parthasarathy, "Identification and control of dynamical systems using neural networks", IEEE Transactions on Neural Networks, Vol.1, No.1, 1990.

[11] K. Narendra, and K. Parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks", Vol.2, No.2, March 1991.

[12] L. Chen, X. Yang and R. Mohler, "Stability analysis of bilinear systems", IEEE Transaction on Automatic Control, Vol.36, No.11, 1991.

[13] M. Nagai, A. Moran, Y. Tamura and S. Koizumi, "Identification and control of nonlinear active pneumatic suspension for railway vehicles using neural networks", Control Engineering Practice, Vol.5, No.8, August 1997.

[14] A. Moran, J. Razuri, D. Sundgren, and R. Rahmani, "Autonomous motion of mobile robot using fuzzy-neural networks", 12th Mexican International Conference on Artificial Intelligence MICAI 2013, Mexico, November 2013.

[15] A. Moran, "Autonomous path following of truck-trailer vehicles using linear-fuzzy control", 3rd International Conference on Control, Automation and Robotics, ICCAR 2017, Nagoya, Japan, April, 2017.