SVM-Based Tree-Type Neural Networks as a Critic in Adaptive Critic Designs for Control

Alok Kanti Deb, *Student Member, IEEE*, Jayadeva, *Senior Member, IEEE*, Madan Gopal, and Suresh Chandra

***Abstract*—In this paper, we use the approach of adaptive critic design (ACD) for control, specifically, the action-dependent heuristic dynamic programming (ADHDP) method. A least squares support vector machine (SVM) regressor has been used for generating the control actions, while an SVM-based tree-type neural network (NN) is used as the critic. After a failure occurs, the critic and action are retrained in tandem using the failure data. Failure data is binary classification data, where the number of failure states are very few as compared to the number of no-failure states. The difficulty of conventional multilayer feedforward NNs in learning this type of classification data has been overcome by using the SVM-based tree-type NN, which due to its feature to add neurons to learn misclassified data, has the capability to learn any binary classification data without *a priori* choice of the number of neurons or the structure of the network. The capability of the trained controller to handle unforeseen situations is demonstrated.**

***Index Terms*—Adaptive control, adaptive critic designs (ACDs), intelligent control, inverted pendulum, linear programming, neural network (NN) applications, support vector machines (SVMs).**

I. Introduction

OPEN-LOOP applications such as signal processing or system identification are significantly different from closed-loop control applications. In the later situation, interactions between the plant dynamics and controller parameters come into play, introducing additional complexity into the system. Controllers should have the essential property of function approximation, and neural networks (NNs) are known to possess good function approximation properties [1], [2]. Recent developments in the application of nonlinear models based on artificial neural networks (ANNs) hold much promise in the field of model-based control, where input–output training data is available, and supervised learning schemes can be used. However, it is desirable to apply more advanced learning methods and intelligent features in control applications that may eliminate the need for analytic modeling of a plant. One approach in this direction is the use of reinforcement learning, where the available information is not as direct, immediate or informative as in supervised learning, and serves more to evaluate the system, as shown in Fig. 1. This is particularly useful for control problems, which require selecting actions whose consequences emerge over uncertain periods, for which input–output training data is not readily available. One class of reinforcement learning topology is the adaptive critic [3]–[6] methodology, which is based on dynamic programming.

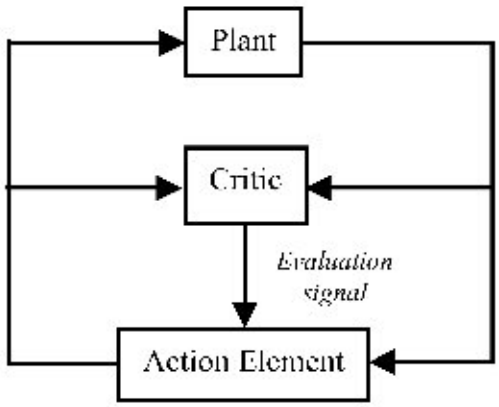

Fig. 1. Reinforcement learning scheme.

Building the critic requires learning highly unbalanced data sets. This is because relevant failure instances may be few, and in NN-based learning approaches, such exemplars may get treated as outliers and be swamped by exemplars of the other class. In the literature, difficulties have been reported with learning binary classification data sets comprising a failure state and several nonfailure states by a multilayer feedforward NN [7]. The problems arose due to the sudden jump in the output value of a failure state from the nonfailure state. The tree-type network that we have used in this paper guarantees complete learning of any binary classification data [8], and has, therefore, been used to overcome this hurdle.

As pointed out, a good critic should be able to generalize well from a few examples, such as from a small number of failure states. Approaches such as ANNs trained with error backpropagation try to minimize the mean squared error over training patterns. However, the error minimization objective does not address maximizing generalization or the classification margin. Therefore, we use support vector machines (SVMs) for the critic because they are known for maximizing generalization while simultaneously minimizing the classification error. The SVM-based tree-type network offers several advantages when used as

Manuscript received December 1, 2005; revised September 20, 2006; accepted February 5, 2007. The work of A. K. Deb was supported by the Indian Institute of Technology (IIT Delhi) through the Ph.D. fellowship.

A. K. Deb was with the Department of Electrical Engineering, Indian Institute of Technology (IIT), New Delhi 110016, India. He is now with the Center for Soft Computing Research: A National Facility, Indian Statistical Institute, Kolkata 700108. India (e-mail: alokkanti@gmail.com).

Jayadeva and M. Gopal are with the Department of Electrical Engineering, Indian Institute of Technology (IIT), New Delhi, 110016, India (e-mail: jayadeva@ee.iitd.ac.in; mgopal@ee.iitd.ac.in).

S. Chandra is with the Department of Mathematics, Indian Institute of Technology (IIT), New Delhi 110016, India (e-mail: chandras@maths.iitd.ac.in).

Digital Object Identifier 10.1109/TNN.2007.899255

the critic. Given any training set, the network will find a solution that learns the entire training set without getting stuck in a local minimum.

When SVMs are applied to nonlinearly separable problems, a kernel function needs to be chosen. The SVM-based tree-type network learns a piecewise linear decision boundary, obviating the need to choose a kernel or kernel parameters. Keeping these points in mind, we have used SVMs as a component of the critic in our adaptive critic design (ACD)-based approach. Finally, the tree may be more amenable to being interpreted in the form of a set of sequential decisions or rules, though this aspect is beyond the scope of this paper.

Conventional SVM formulations such as [9]–[11] optimize a quadratic objective function subject to certain linear inequality constraints. Another class of SVMs, termed as least squares support vector machines (LS-SVMs) has been proposed in [12] and [13], where the constraints are of the equality type. In LS-SVMs, the classifier or regressor is obtained by solving a set of linear equations. Using a chosen kernel, this technique gives a good estimation by placing the kernel functions at the training points. In this paper, an LS-SVM regressor has been trained to act as the controller. The performance of the controller is indicated by a "critic" in the form of a constructive network [14], developed by using the linear programming (LP) framework, each of whose nodes acts as a maximum-margin type support vector classifier.

The rest of this paper is organized as follows. Section II briefly introduces the different ACDs. Section III describes two learning structures: the SVM-based tree-type NN, and LS-SVMs. Section IV describes the control strategy adopted in this work. Section V discusses experimental results. Section VI contains concluding remarks.

II. ACDs

ACDs have evoked growing interest in nonlinear control, because of their capability to approximate optimal control over time in nonlinear environments [3]. Any real-life optimization problem can be cast as a minimization or a maximization task, and an optimal solution can be achieved if dynamic programming is used. However, dynamic programming methods are computation-intensive, due to initiation of the search process from the goal, and the dependence at any stage on the computation of the cost-to-go to the next stage for different available controls. They also need highly efficient data structures and programming capabilities to carry out the method stage-by-stage. The "curse of dimensionality" has limited their use as a tool for finding solutions to complex optimization problems. Attempts have been made to overcome this curse by using heuristic strategies to approximate the cost function. ACD offers one such approach, where it is attempted to build a "critic" by using a function approximation structure that learns to approximate the cost function in dynamic programming.

The three basic methods of ACDs are heuristic dynamic programming (HDP), dual heuristic programming (DHP), and globalized dual heuristic programming (GDHP). The basic blocks of an ACD are the action element and the critic element (evaluation module), but in some of the structures a model network is also used for system identification. If the critic element has the action/control signals as part of its input, those designs are referred to as action-dependent ACDs (ADACDs). Different ACDs have been illustrated with examples in [4]–[6].

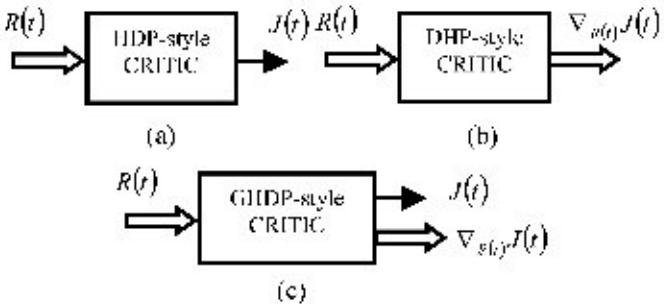

Fig. 2. Different ACDs. (a) Critic for HDP. (b) Critic for DHP. (c) Critic for GDHP.

In a discrete-time nonlinear dynamical system

$$x(t+1) = P[x(t), u(t), t] \tag{1}$$

where $x(t) \in \Re^n$ denotes the system state and $u(t) \in \Re^m$ denotes the control action, the system changes at every instant with the application of control inputs. This change is guided by the dynamical nature of the system, which is captured by the function P. As the system's state changes from instant to instant, any instant is associated with a cumulative cost to drive the system from that instant to the goal. The performance index $J(\cdot)$ for HDP at any instant is given by

$$J[x(t), t] = \sum_{k=t}^{\infty} \gamma^{k-t} U[x(k), k] \tag{2}$$

where $U(\cdot)$ is the utility function or local cost function, and γ is the discount factor, with $0 < \gamma \leq 1$. The aforementioned is the cost-to-go of the state $x(t)$ for the infinite horizon problem. If the control actions $u(t)$ are also part of the utility function, we have the action-dependent heuristic dynamic programming (ADHDP), whose performance index is given by

$$J[x(t), t] = \sum_{k=t}^{\infty} \gamma^{k-t} U[x(k), u(k), k]. \tag{3}$$

Fig. 2(a) shows the critic for the HDP/ADHDP scheme. In HDP, $R(t)$ is the vector of states $x(t)$ of the plant, while for ADHDP, it is a concatenation of the states $x(t)$ and the control vector $u(t)$. Minimization of $J(t)$ is the desired objective. Equations (2) or (3) may be more concisely written as

$$J[x(t), t] = \sum_{k=t}^{\infty} \gamma^{k-t} U[k]. \tag{4}$$

Expanding the previous, and using the terminology for the cost-to-go at $(t+1)$, we obtain

$$\begin{aligned} J[t] &= U[t] + \gamma U[t+1] + \gamma^2 U[t+2] - \dots \\ &= U[t] + \gamma J[t+1]. \end{aligned} \tag{5}$$

At any instant "t," the objective is to find an appropriate control sequence $u(k)$, $k = t, t+1, \dots$, so that the function J is minimized. The task of the critic network is to learn the function

$J(t)$. In general, the critic elements are trained by minimizing a certain error measure over time, say

$$\|E_1\| = \sum_t E_1^2(t) \tag{6}$$

where, following [5] and [6], we have $E_1(t) = J(t) - U(t) - \gamma J(t+1)$.

Any function approximating structure may be trained to act as the critic, so that at any instant it can output the cost function for the immediate future. If the critic is parameterized as W_c, the input–output relationship of the critic network in an ADHDP may be written as

$$J[t] = J\left[x(t), u(t), W_c\right]. \tag{7}$$

In DHP and action-dependent DHP (ADDHP), the critic network estimates the derivatives of J with respect to the vector $R(t)$, where $R(t)$ is the vector of states $x(t)$ of the plant (DHP), or a concatenation of the states $x(t)$ and the control vector $u(t)$ (ADDHP), as shown in Fig. 2(b). In general, the critic element is trained to minimize a certain time-dependent error measure

$$\|E_2\| = \sum_t E_2^T(t) E_2(t) \tag{8}$$

where, following [5] and [6], we have

$$E_2(t) = \nabla_{R(t)} J(t) - \gamma \nabla_{R(t)} J(t-1) - \nabla_{R(t)} U(t). \tag{9}$$

Here, the derivatives are taken with respect to the components of the vector $R(t)$. The need for the existence of derivatives in the error measure, and the training of the critic by minimizing a certain norm of such an error measure, complicates its case as compared to HDP.

GDHP and its AD forms are a fusion of HDP and DHP where the critic network adapts in such a way that it estimates two components: a scalar component J and its derivatives with respect to the components of $R(t)$. The vector $R(t)$ is a vector of the states $x(t)$ of the plant in the case of GDHP, or a concatenation of the states $x(t)$ and the control vector $u(t)$ in the case of action-dependent GDHP (ADGDHP), as shown in Fig. 2(c). Therefore, the adaptation of a GDHP critic would involve minimizing an error measure that is equal to the sum of the error measures of HDP and DHP, as given by (6) and (8), respectively. GDHP and ADGDHP are the most complicated ACD approaches and are expected to be superior to other methods [5], [6].

III. Learning Structures

A. SVM-based Tree-Type NNs

SVMs are known for their pattern classification capabilities [9]–[11]. The most popular SVM, "the maximum margin classifier," aims at minimizing an upper bound on the generalization error through maximizing the margin between two disjoint half-spaces. These may be in the original input space for a linear classification problem, or in a higher dimensional feature space for a nonlinear classification problem. Given a training set, $\{S = (\boldsymbol{x}^i, y^i) : \boldsymbol{x}^i \in \Re^n; y^i \in \{0,1\} : i \in I : \text{card}(I) = N\}$, which is linearly separable, the hyperplane $\boldsymbol{w}$ that solves the optimization problem

$$\text{(QPP)} \quad \min_{w,w_0} \langle \boldsymbol{w} \cdot \boldsymbol{w} \rangle \tag{10}$$

subject to

$$y^i \left(\langle \boldsymbol{w} \cdot \boldsymbol{x}^i \rangle + w_0 \right) \geq 1, \qquad i = 1, 2, \ldots, N \tag{11}$$

realizes the maximum margin hyperplane [9]–[11] with geometric margin $\gamma = (1/\|\boldsymbol{w}\|_2)$. On the other hand, for the task of binary classification of the set S, the goal is to obtain the weight vector $\begin{bmatrix} w_0 \\ \boldsymbol{w} \end{bmatrix}$, such that

$$\sum_{d=0}^{n} w_d x_d^i = \begin{cases} > \Delta, & \text{if } y^i = 1 \\ \leq -\Delta, & \text{if } y^i = 0 \end{cases}; \qquad x_0^i = 1; \qquad i \in I \tag{12}$$

where i denotes the ith sample, I is the index set of the training data set with cardinality N, and Δ is a positive quantity introduced for better separation of the data. Samples from the data set having output $y^i = 1$ belong to Class 1, while samples having output $y^i = 0$ belong to Class 0. Using SVMs, the binary classification problem can be solved by learning a tree-type NN [14]. This network grows by adding one neuron at a time. Given an input–output data set, a perceptron tries to satisfy all the inequalities in (12) for each input. Following [15], the inequalities in (12) can be reduced to the following equalities:

$$\sum_{d=0}^{n} \left(w_d^{(-)} - w_d^{(-)} \right) x_d^i - \left(s_i^{(+)} - s_i^{(-)} \right) = \Delta, \qquad \text{if } i \in I_1 \tag{13a}$$

$$\sum_{d=0}^{n} \left(w_d^{(-)} - w_d^{(-)} \right) x_d^j + \left(s_j^{(+)} - s_j^{(-)} \right) = -\Delta, \qquad \text{if } j \in I_0 \tag{13b}$$

where $w_i^{(+)}$, $w_i^{(-)}$, $s_i^{(+)}$, $s_i^{(-)}$, $s_j^{(+)}$, and $s_j^{(-)}$ are nonnegative variables; I_1 is the index set of data samples of Class 1 with cardinality m; I_0 is the index set of data samples of Class 0 with cardinality k; $I = I_1 \cup I_0$; $I_1 \cap I_0 = \phi$; and $x_0^i = x_0^j = 1$. Following [15], any weight w_d can be represented as $w_d = w_d^{(+)} - w_d^{(-)}$, where $w_d^{(+)}$ and $w_d^{(-)}$ are nonnegative variables. Hence, w_d can take on either a positive or negative value depending on the values of $w_d^{(+)}$ and $w_d^{(-)}$. The surplus and slack variables $s_i^{(+)}$ and $s_i^{(-)}$ are used to convert the inequalities in (12) to equalities [see (13a) or (13b)].

Any constraint from the training set (say, the pth one) is satisfied if the corresponding slack variable

$$s_p^{(\cdot)} = 0, \qquad \text{for} \quad p \in I \tag{14}$$

where I is the index set of the training data set with cardinality N. Therefore, given a set of input–output data pairs, the obvious objective is to minimize the number of inequalities that are not

TABLE I
TYPES OF LPPS

Type	Value of λ and μ	Case
Type I	$\lambda = 1; \mu = 1$	substantial value of m and k
Type II	$\lambda = \frac{1}{m}; \mu = \frac{1}{k}$	substantial value of m and k
Type III	$\lambda = M; \mu = \frac{1}{k}; M \geq 1$	$m \ll k$
Type IV	$\lambda = \frac{1}{m}; \mu = M; M \geq 1$	$k \ll m$

TABLE II
CLASSIFICATION OF SAMPLES BASED ON OUTPUT

Class	Actual Output	Desired Output	No. of Samples
C_1	0	0	N_1
C_2	1	1	N_2
C_3	0	1	N_3
C_4	1	0	N_4

satisfied, which can be formulated as the following linear programming problem (LPP) [8]:

$$\text{(LPP)} \quad \min \sum_{i=1}^{m} \lambda s_i^{(-)} + \sum_{j=1}^{k} \mu s_j^{(-)} \tag{15}$$

subject to the constraints (13a) and (13b). Given a binary classification data set, one may have four types of LPPs, as shown in Table I. These four types are obtained by changing the relative weightings of slack variables $s_i^{(-)}$ and $s_j^{(-)}$ in the objective function (15) in terms of the coefficients λ and μ.

The different LPP types lead to different solutions for a binary classification problem. Type I is a trivial choice of the coefficients of the objective function and Type II leads to quick convergence. However, for skewed data sets, where the number of samples of one class is much smaller than that of the other class, using a Type II LPP leads to a trivial solution. For skewed data sets, Types III and IV LPPs have been proposed with, respectively, higher values of λ and μ, so as to increase the possibility of elimination of the corresponding slack variable from the basis during early runs of the Simplex algorithm.

In Table I, M represents the value of λ or μ that is used for the coefficients of the slack variables $s_i^{(-)}$ and $s_j^{(-)}$, when the objective function (15) is used for a Type III or Type IV LPP, respectively. This has been necessitated to overcome the learning problem when the number of samples of one of the two classes is much smaller than that of the other. Type III LPPs are used when the number of samples of Class 1 is much lower than that of Class 0. In such a case, λ and μ are chosen such that a higher value of λ ($\geq M$) is used, so as to assign greater importance to the slack variables $s_i^{(-)}$ corresponding to patterns of Class 1. This increases the possibility of elimination of these variables from the basis set during early runs of the Simplex algorithm. LPPs of Type IV also behave the same way for the case

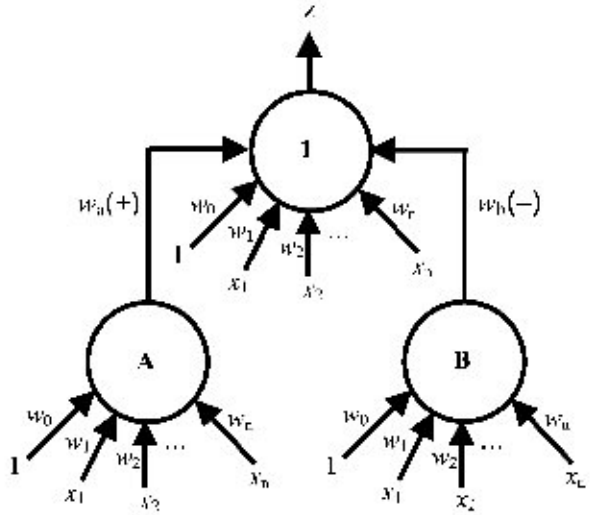

Fig. 3. Parent neuron and its two child neurons, A and B.

TABLE III
REQUIRED OUTPUT OF NEURON A AND NEURON B

Class	Output of A	Output of B
C_1	0	Don't care
C_2	Don't care	0
C_3	1	0
C_4	0	1

where the training set samples of Class 0 are much fewer than the number of Class 1 samples.

A hyperplane is chosen from the possible LPP solutions, as in Table I, based on the performance of the hyperplanes on the training set and the validation set (if present). The chosen hyperplane classifies the N training samples, as in Table II, based on the output. Here, $N_1 + N_2 + N_3 + N_4 = N$, which is the total number of samples.

Any sample belonging to class C_3 or C_1 has an incorrect output. Child neurons A and B are added to correct for such samples, as shown in Fig. 3. Neuron A provides a positive input, while neuron B effectively provides a negative input, due to the negative weight $w_b(-)$. The desired outputs of the child neurons are also shown in Table III.

Denoting the outputs of neuron A and B for the ith sample as x_a^i and x_b^i, respectively, the constraints that are to be simultaneously satisfied by the parent neuron, and the desired outputs of A and B together are given by

$$\sum_{d=0}^{n} w_d x_d^i + w_a x_a^i + w_b x_b^i \geq \Delta_i, \qquad \text{if } i \in I_1 \tag{16a}$$

$$\sum_{d=0}^{n} w_d x_d^j - w_a x_a^j + w_b x_b^j \leq -\Delta_j, \qquad \text{if } j \in I_0. \tag{16b}$$

The solution of the maximum margin classification problem applied to the linearly separable data set comprising the parent neuron augmented with the desired inputs from the child neurons, gives the weight vector $\begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}$, as well as weights of the interconnections from neuron A and B, viz. w_a and w_b. Thus,

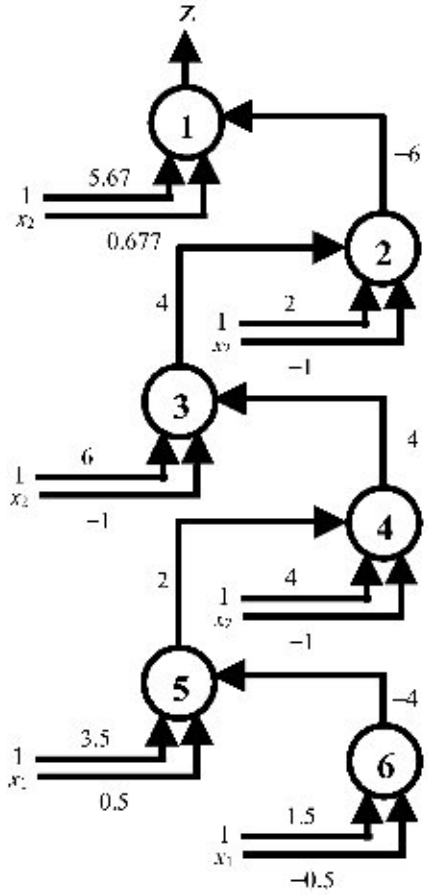

Fig. 4. SVM-based tree-type NN for the synthetic data set.

the growth of the network takes place until a neuron can learn its whole training set without any classification error.

The constructive nature of the SVM-based tree-type NN actually gives rise to a deep structure of the network, the final decision hyperplane being decided at the topmost neuron. Individual neurons down the tree represent a rule structure that learns a part of the training set, while removal of a certain branch/node would lead to unlearning of some of the data at the topmost node. For a given training set, it is a one-shot training of the complete data set, and the network gets fixed. If the training set changes or if a new data is added to an existing training set, a new network has to be trained afresh. The approach gives a pointer in the direction of developing a "critic" by constructive techniques.

Example: Consider a synthetic 2-D data set with samples of Class 1 located at $\{(3,3),(3,7),(5,5),(7,3),(7,7)\}$, and samples belonging to Class 0 located at $\{(1,1),(1,5),(1,10),(9,1),(9,5),(9,10)\}$. Fig. 4 shows how the SVM-based tree-type NN having six perceptrons learns the data set by hierarchically correcting the samples learned incorrectly at the previous layer. Fig. 5 shows the points in 2-D, along with the optimal decision boundaries composed of piecewise linear segments learned by the network.

B. LS-SVMs

A class of SVMs, called LS-SVMs [12], [13], use equality type constraints in their formulation. This allows the solution to

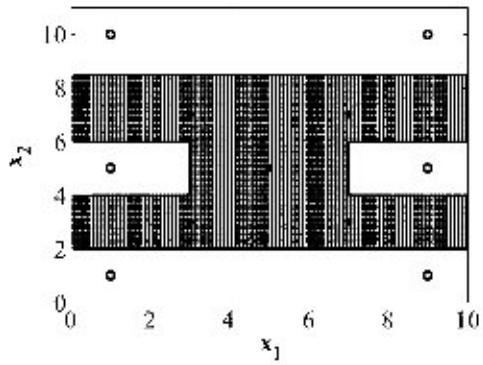

Fig. 5. Classification contours generated by SVM-based tree-type NN for the synthetic data set.

be obtained by solving a set of linear equations, instead of by solving a quadratic programming problem (QPP) as in the case of classical SVMs. LS-SVMs have been applied to the N-stage optimal control problem [16].

Let $S_t = \{(\boldsymbol{x}^i, y^i) : \boldsymbol{x}^i \in \Re^n; y^i \in \Re; i \in I : \mathrm{card}(I) = N\}$ be a given training data set. The nonlinear mapping $\phi : \Re^n \to \Re^{n_h}$ maps the input data into a high-dimensional feature space. In SVMs, the aim is to reduce the empirical risk

$$R_{\mathrm{emp}}(\boldsymbol{w}, w_0) = \frac{1}{N}\sum_{i=1}^{N} \left| y^i - \langle \boldsymbol{w} \cdot \phi(\boldsymbol{x}^i) \rangle - w_0 \right|_\varepsilon \qquad (17)$$

where $\boldsymbol{w} \in \Re^{n_h}$, $w_0 \in \Re$, and $|a|_\varepsilon$ denotes a loss function that penalizes a loss a depending on the parameter ε. Some commonly used loss functions are the ε-insensitive loss function, the quadratic ε-insensitive loss function, and Huber's loss function [9]–[11].

The risk minimization problem given by (17) can be expressed as the following regression problem:

$$\underset{\boldsymbol{w}, w_0, \xi}{\text{Minimize}} \quad \frac{1}{2}\langle \boldsymbol{w} \cdot \boldsymbol{w} \rangle + \gamma \frac{1}{2}\sum_{i=1}^{N} \xi_i^2 \qquad (18)$$

subject to the equality constraints

$$y^i = \langle \boldsymbol{w} \cdot \phi(\boldsymbol{x}^i) \rangle + w_0 + \xi_i, \qquad i = 1, 2, \ldots, N. \qquad (19)$$

Objective functions as in (18) are used so as to minimize both the structural and empirical risks, thereby providing better generalization. For the previous problem, the Lagrangian is given by

$$L_{\mathrm{LS}}(\boldsymbol{w}, w_0, \xi, \alpha) = \frac{1}{2}\langle \boldsymbol{w} \cdot \boldsymbol{w} \rangle + \gamma \frac{1}{2}\sum_{i=1}^{N} \xi_i^2 - \sum_{i=1}^{N} \alpha_i \left(\langle \boldsymbol{w} \cdot \phi(\boldsymbol{x}^i) \rangle + w_0 + \xi_i - y^i \right) \qquad (20)$$

where α_is are the Lagrangian multipliers that can be positive or negative, as follows from the equality constraints of the Karush–Kuhn–Tucker conditions. Following [12] and [13], we have:

$$\nabla_{\boldsymbol{w}} L_{\mathrm{LS}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}^i) \qquad (21a)$$

TABLE IV
SOME KERNEL FUNCTIONS

Kernel	Kernel Function	Parameters
Linear kernel	$K(x^i, x^j) = \langle x^i \cdot x^j \rangle$	
Multilayer Perceptron kernel	$K(x^i, x^j) = \tanh(s\langle x^i \cdot x^j \rangle + t^2)$	t :bias s :scale parameter
Polynomial kernel	$K(x^i, x^j) = (\langle x^i \cdot x^j \rangle + t)^d$	t :intercept d :degree of the polynomial
Gaussian kernel	$K(x^i, x^j) = e^{\frac{(x^i - x^j)(x^i - x^j)}{\sigma^2}}$	σ^2 :variance

$$\frac{\partial L_{LS}}{\partial w_0} = 0 \Rightarrow \sum_{i=1}^{N} \alpha_i = 0 \tag{21b}$$

$$\frac{\partial L_{LS}}{\partial \xi_i} = 0 \Rightarrow \alpha_i = \gamma \xi_i, \qquad i = 1, 2, \ldots, N \tag{21c}$$

$$\frac{\partial L_{LS}}{\partial \alpha_i} = 0 \Rightarrow \langle \boldsymbol{w} \cdot \phi(\boldsymbol{x}^i) \rangle + w_0 + \xi_i - y^i = 0, \quad i = 1, 2, \ldots, N. \tag{21d}$$

Substituting $\boldsymbol{w}$ and ξ_i from (21a) and (21c) in (21d), we can write (21b) and (21d) compactly as

$$\begin{bmatrix} 0 & \vec{1}^T \\ \vec{1} & \mathbf{K} + \gamma^{-1} I \end{bmatrix} \begin{bmatrix} w_0 \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix} \tag{22}$$

where $y = [y_1\ y_2\ \ldots\ y_N]^T$; $\vec{1} = [1\ 1\ \ldots\ 1]_{1 \times N}]^T$; $\boldsymbol{K}_{ij} = K(\boldsymbol{x}^i, \boldsymbol{x}^j) = \langle \phi(\boldsymbol{x}^i) \cdot \phi(\boldsymbol{x}^j) \rangle$, $i, j = 1, 2, \ldots, N$; and $\boldsymbol{\alpha} = [\alpha_1\ \alpha_2\ \ldots\ \alpha_N]^T$. Some commonly used kernel functions are mentioned in Table IV.

As is evident from (21c), for the least squares (LS) case, the Lagrangian variables α_i are proportional to the errors at the data, contrary to classical SVMs, where nonzero α_is correspond to the support vectors. Hence, sparseness is lost in the LS case. With some choice of the kernel matrix $\boldsymbol{K}$ and the regularization parameter γ, (22) represents a set of linear equations that can easily be solved for w_0 and α. Here lies the advantage of using LS-SVMs, where one solves a set of linear equations, instead of a QPP, as in classical SVMs. The resulting LS-SVM model for function approximation becomes

$$y(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i K(\boldsymbol{x}^i, \boldsymbol{x}) + w_0. \tag{23}$$

IV. CONTROL STRATEGY

It is desired to develop a controller for a plant so as to minimize the occurrence of failures, and also to provide a smooth operation. To achieve this control objective, the ADHDP strategy has been adopted, whereby the critic and action networks work in tandem; the critic network tracks the performance of the plant from no-failure to failure conditions, while the actual control inputs are generated by the action network based on the plant state. The quality of control inputs generated by the action network is indicated by binary signals generated at the output of the critic network. The complete scheme for an ADHDP-based controller is shown in Fig. 6.

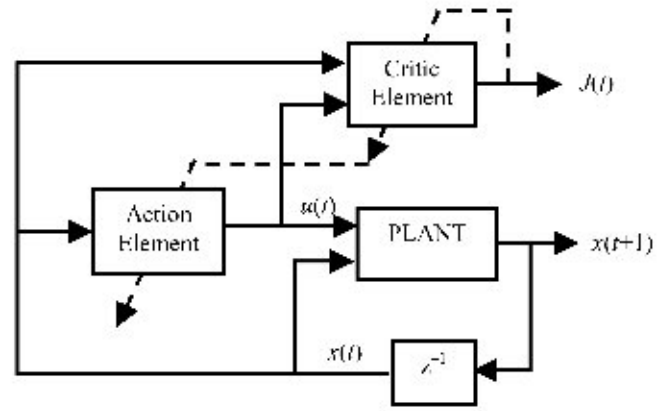

Fig. 6. Control using ADHDP.

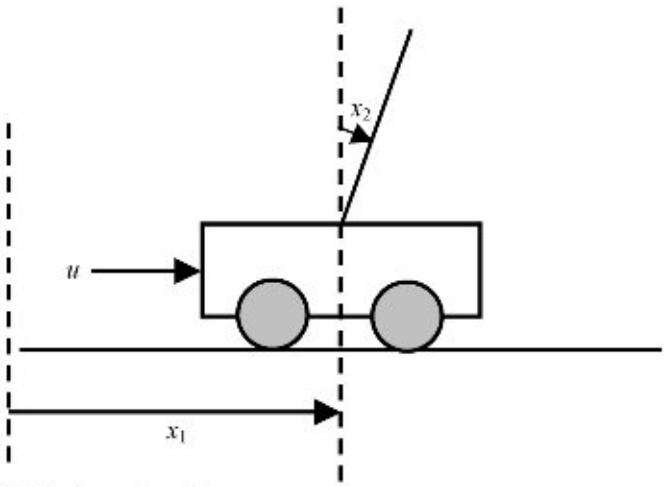

Fig. 7. Cart-pole model.

A popular example of a plant where a controller of this type may be incorporated is the inverted pendulum or the cart-pole problem, as shown in Fig. 7. The objective here is to balance the pole in an upright position. The bottom of the pole is hinged on a cart. The only control input available is the horizontal force that can be applied to the cart along the track. Using this failure condition, the utility function of the cart-pole system is given by

$$U(t) \begin{cases} 0, & \text{when no failure} \\ 1, & \text{failure condition} \end{cases}. \tag{24}$$

Due to the formulation of the utility function as aforementioned, the reinforcement signal is obtained only when the controller is unable to control the plant. This information regarding the smooth operation to a failure condition of the plant's states is used for retraining the critic and action networks for an ADHDP-based design. The plant states and the corresponding utility value of 0/1 depending upon no-failure/failure constitute the data for a binary classification problem. This data can be used for training a learning machine as the critic.

Training of an ACD is a sequential process, with the training alternating between the critic and action networks. The critic network is first trained with the failure data. Starting with a single node, the SVM-based tree-type NN adds up to two child

neurons that provide corrections for samples that are misclassified by the parent node. Learning at the child neurons proceeds in a similar fashion. However, the number of patterns that need to be learned at the child neurons is less than that at the parent, since the parent will correctly classify a finite number of training patterns. Consequently, the number of misclassified patterns reduces at each step, and becomes zero after a finite depth of the tree. The critic's training is stopped when it has learned the complete training set comprising the no-failure and failure states.

Once the critic is trained, action training starts. It is desired to minimize $J(t)$ at the output of the critic in the immediate future, so that it would optimize the overall cost expressed as the sum of future $U(t)$. This will necessitate training of the action element, so that the output of the critic is as small as possible. Ideally, one would like to have the output of the critic network equal to zero, so that the performance index J is minimized. The desired mapping needed for training the action network in the present ADHDP set up is given by

$$A : \{\boldsymbol{x}(t)\} \rightarrow \{\mathbf{0}(t)\} \tag{25}$$

where $\{\mathbf{0}(t)\}$ indicates that the target value at the output of the critic network is zero for an input $\{\boldsymbol{x}(t)\}$ at the input of the action network. In other words, at instant "t," the action network should generate appropriate control actions $\{u(t)\}$ in response to the state $\{\boldsymbol{x}(t)\}$ so as to drive a certain $J(t)$ to zero.

V. Experimental Results: Control by ADHDP

A. Plant Description

To test the previous control strategy, the chosen plant was a single-input–multiple-output (SIMO) system, viz. the model of a single link inverted pendulum. The inverted pendulum case study was chosen as it is a benchmark application in the control literature, and its principle underlies some more complex control systems like balancing the booster of a rocket during launch, or multiple links of a robotic arm. The model and its parameters have been adopted from an actual experimental test bed designed by Feedback Instruments (Crowborough, East Sussex, U.K., [17]. The state vector of the system is given by

$$\boldsymbol{x} = [x_1 \quad x_2 \quad x_3 \quad x_4]^T \tag{26}$$

where x_1 is the cart position (measured from the center of the rail); x_2 is the angle between the upward vertical and the axis of the pole, measured counterclockwise from the cart; and x_3 is the cart velocity and x_4 is the pendulum angular velocity.

The pendulum can rotate in the vertical plane, while the cart can move on a horizontal rail lying in the plane of rotation. The only control that can be applied is in terms of a horizontal force u applied on the cart parallel to the rail. The dynamical equations governing the plant motion [17] are given by

$$\dot{x}_1 = x_3 \tag{27a}$$

$$\dot{x}_3 = \frac{\alpha\left(u - T_c - \mu x_4^2 \sin x_2\right) + l\cos x_2\left(\mu g \sin x_2 - f_p x_4\right)}{J - \mu l \sin^2 x_2} \tag{27b}$$

TABLE V
Parameters of the Pendulum-Cart Setup [17]

Symbol	Parameter	Value
	Track limits	± 0.5m
g	Acceleration due to gravity	9.81 m/s^2
l	Distance between mass center and the axis of rotation	0.017 m
m_c	Mass of the cart	1.12 kg
m_p	Mass of the pole	0.11 kg
M	Magnitude of the control force	17.0 N
J	Moment of inertia of the system	0.0136 kg-m^2
f_p	Friction coefficient of pole rotation	Negligible
f_c	Friction coefficient of cart	0.05 Ns/m

$$\dot{x}_2 = x_4 \tag{27c}$$

$$\dot{x}_4 = \frac{l\cos x_2\left(u - T_c - \mu x_4^2 \sin x_2\right) + \mu g \sin x_2 - f_p x_4}{J + \mu l \sin^2 x_2} \tag{27d}$$

where $\alpha = l^2 + (J/(m_c + m_p))$; $\mu = (m_c + m_p)/l$; g is the acceleration due to gravity; m_c is the mass of the cart; m_p is the mass of the pendulum; l is the distance from the axis of rotation to the center of mass of the pendulum-cart system; J is the moment of inertia of the pendulum-cart system with respect to the center of mass; T_c denotes the friction in the motion of the cart, where $T_c = f_c x_3$; and D_p is the moment of friction in the angular axis of the pendulum, proportional to the angular velocity, i.e., $D_p = f_p x_4$. The admissible controls are bounded, i.e., $|u(t)| < M$. The different parameter values of the pendulum-cart setup are given in Table V.

B. Control Setting

Several ADHDP strategies have been attempted for control applications. Developing such a control strategy involves designing a controller ("action" element) and a module to evaluate the controlling function ("critic" element). The modules need to have function approximation capabilities. Making use of their function approximation capabilities, NNs have been used to develop these functional blocks [5], [7], [18], [19]. Another approach attempts to make use of the knowledge used by experienced operators in approximate linguistic terms for developing these functional blocks by fuzzy NNs [20], [21]. NNs are plagued by the need for a large number of training examples, occurrence of local minima, oscillations, and "forgetting" problems, while fuzzy-logic-based approaches need a detailed design methodology of correctly deciding the structure, variables, and its membership functions, choice of the rule base and tuning of the parameters. The present investigation adopts a constructive approach in the ADHDP-based strategy.

1) Critic Element: Learning Problem: As discussed in Section IV, in the control approach using ADHDP-based design, a batch of failure data comprising the plant states and the corresponding utility values of 0/1 (depending upon no-failure/failure of the plant) constitute a binary classification data set. Since failure may occur after healthy running of a

TABLE VI
MAXIMUM AND MINIMUM VALUES USED FOR NORMALIZATION OF THE STATE VARIABLES OF THE PENDULUM-CART SYSTEM

Symbol	State Variable	Maximum	Minimum
$\theta(t)$	Pendulum Angle	$\pi/15$ rad	$-\pi/15$ rad
$\dot{\theta}(t)$	Pendulum Angular Velocity	2π rad/s	-2π rad/s
$s(t)$	Cart Position	0.5 m	0.5 m
$\dot{s}(t)$	Cart Velocity	3 m/s	-3 m/s

plant, it is a "rare event." The difficulty of learning such data using multilayer NNs with nonlinear activation functions and backpropagation training has been reported in [7], as the state value suddenly changes from a no-failure state to a failure one.

Conventional NN learning methods employ iterative schemes for updating the weights. In general, the incremental change in weight connecting a jth neuron to the ith neuron of the next layer is given by

$$\Delta w_{ij} \quad -\eta \frac{\partial E}{\partial w_{ij}} \tag{28}$$

where η is the learning rate and E is the sum-squared error between the target and actual output at the network output.

The most popular backpropagation algorithm [22], [23] computes the sum-squared error E at the network's output and updates the weights layerwise. Convergence of algorithms such as "backprop" and its variants depends on the initial weights and the choice of various parameters of the algorithm. In general, classical derivative-based weight updating algorithms are plagued by the need for a large number of training samples, occurrence of local minima, oscillations, and "forgetting" problems. The problems and performance sensitivity of multilayer feedforward networks to the number of hidden units, and difficulties due to the occurrence of local minima have been shown in [24].

Although it works well for a wide range of problems, for the particular case of learning binary mappings, gradient-based training of feedforward NNs using E as the cost function with nonlinear activation functions like the "sigmoid" and "tanh," often gets stuck with some outputs wrongly learned [25]. If $f(\cdot)$ denotes the activation value of a neuron, the incremental change in weight in a multilayer network by backpropagation training follows the proportional relationship:

$$\Delta w_{ij} \propto f_i'(\cdot). \tag{29}$$

The terms $f'(\cdot) = f(\cdot)[1 - f(\cdot)]$ for "sigmoid" and $f'(\cdot) = (1/2)[1 - \{f(\cdot)\}^2]$ for "tanh" tend to zero as the activation value saturates to 0/1 and $-1/1$, respectively. Thus, it is theoretically possible for the network to get stuck with some weights not getting updated. When using multilayer NNs with nonlinear activation functions like "tanh" or "sigmoid," if most of the samples of a particular class are learned, then the network tends not to learn samples of the other class [25]. If such a network is used as the critic, it tends not to learn the failure state that is a "rare event" [7]. In error backpropagation-based ANNs, such exemplars may get treated as outliers and may be swamped by exemplars of the other class. This common phenomenon has been discussed at length in [25], where the use of evolutionary strategies has been suggested. Error minimization does not address maximizing generalization or the classification margin. Therefore, we use SVMs for the critic because they are known for maximizing generalization while simultaneously minimizing the classification error.

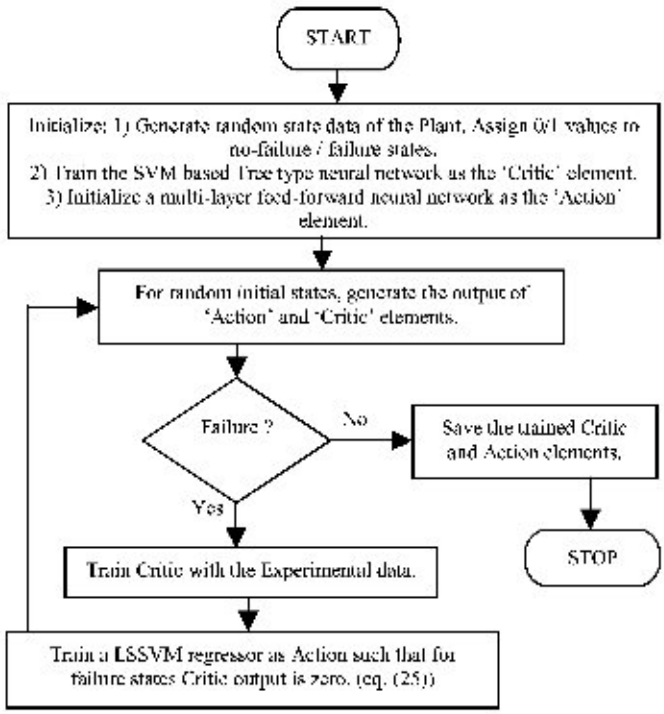

Fig. 8. Flowchart showing ADHDP-based controller training. Critic element: SVM-based tree-type NN. Action element: LS-SVM regressor.

Our Approach: This work attempts to overcome the aforementioned difficulty in learning binary classification data; this is achieved by using the SVM-based tree-type NN [14] as the critic. The approach is based on the interesting set of NN learning methods that uses LP-based formulations [15], [26]–[31] to determine network weights as described in Section III-A. The constructive nature of this network, based on LP formulations of the problem, guarantees complete learning of any binary classification data, as has been demonstrated on real benchmark data sets [8].

2) Action Element: The action element is the main controller that generates control actions depending on the plant states. Making use of the generalization capabilities of SVMs, an LS-SVM regressor with a radial basis function (RBF) kernel was used as the action element. Consequently, the number of hidden units did not need to be determined *a priori*, and centers do not have to be specified for the Gaussian kernels, as the kernels are centered on the failure states. The regularization parameter γ and the kernel parameter σ^2 were tuned to minimize error during training.

3) Plant States: If the displacement of the cart and the angular displacement of the pole are denoted by $s(t)$ and $\theta(t)$, respectively, then the four components of the plant's states that have been considered in the control scheme are $x_1 \quad s(t)$, $x_2 = \theta(t)$, $x_3 = \dot{s}(t)$, and $x_4 = \dot{\theta}(t)$.

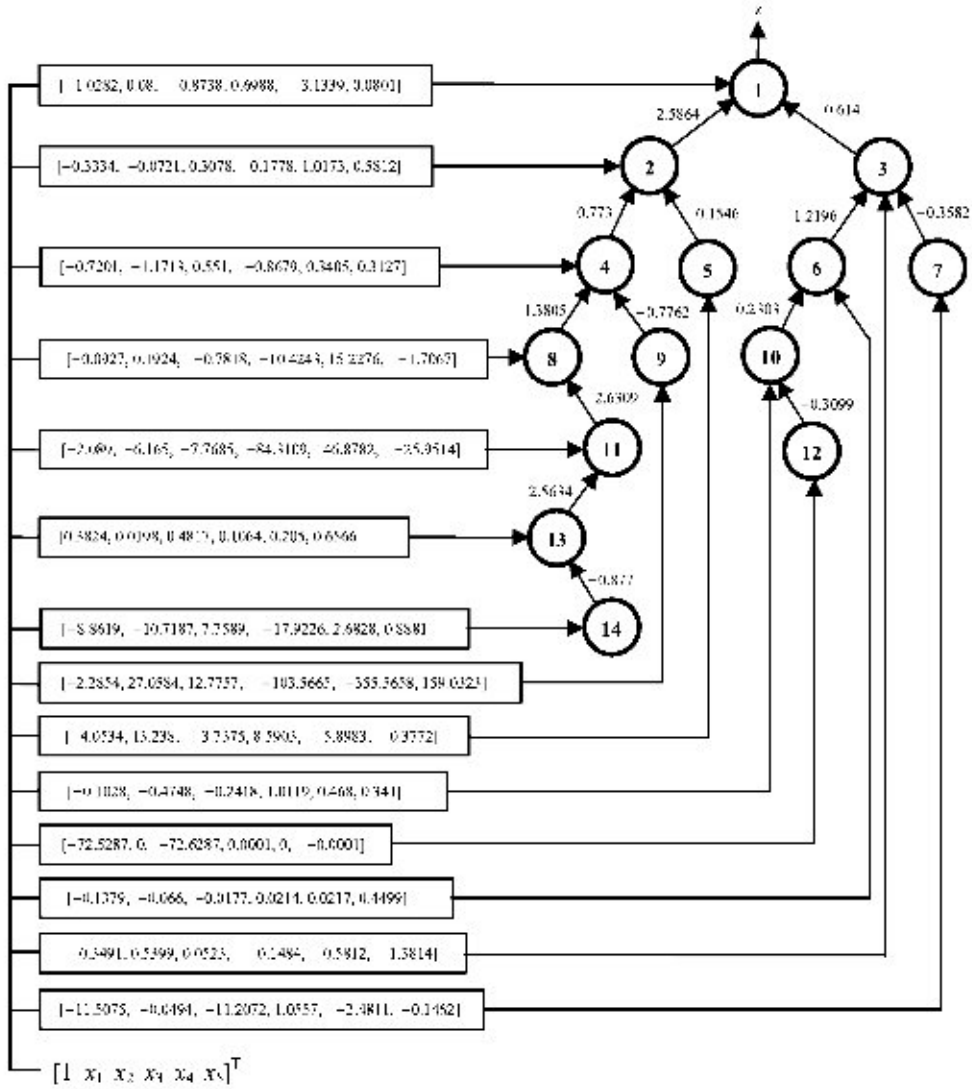

Fig. 9. Structure of the SVM-based tree-type NN that learns to be the "critic" in the ADHDP control strategy for the inverted pendulum problem. The input vector has five components. The weights indicated in the boxes contain six corresponding components, viz. w_0, w_1, w_2, w_3, w_4, and w_5. The first weight w_0 is the weight corresponding to the constant input.

4) Utility Function: The actual utility function used for developing the ADHDP-type control scheme is given by

$$U(t) = \begin{cases} 0, & |\theta(t)| \leq 12^\circ \text{ and } |s(t)| \leq 0.5 \text{ m} \\ 1, & \text{otherwise} \end{cases}. \qquad (30)$$

The utility function penalizes a state if the pole angle falls past 12° on either side of the vertical, or if the cart reaches the end of the track 0.5 m from the center. For the plant being considered, the state variable of utmost interest is the pendulum vertical angle $\theta(t)$, which renders the plant to the "failure" state most often.

C. Controller Implementation

Using the plant parameters as given in Table V, the complete control scheme was built by using the simulation software SIMULINK (version 5.0) and by interfacing it with MATLAB [version 6.5(R13)] programming environment. The plant states and the corresponding critic output are available in the MATLAB workspace to retrain the action element after obtaining the failure data.

In our implementation, all the state variables and the input u have been normalized to lie between -1 and 1. The maximum and minimum values used for normalizing the state variables of the pendulum-cart system are shown in Table VI.

The states and the input of the critic and the action networks were randomly initialized. The initialized states and the input give the initial training set for the action element. Initial training data was generated by assuming different random positions of the pendulum over its two angular limits, and also over the length of the track. A total of 168 states were initialized. The experiments were performed using this initialization to obtain reliable experimental data. A three-layer feedforward network was used as the initial action element, which during subsequent training with actual fault data, was replaced by the LS-SVM regressor. Applying the utility function on the randomly initialized states, the utility value of each state x can be obtained. The initial states, control inputs, and the

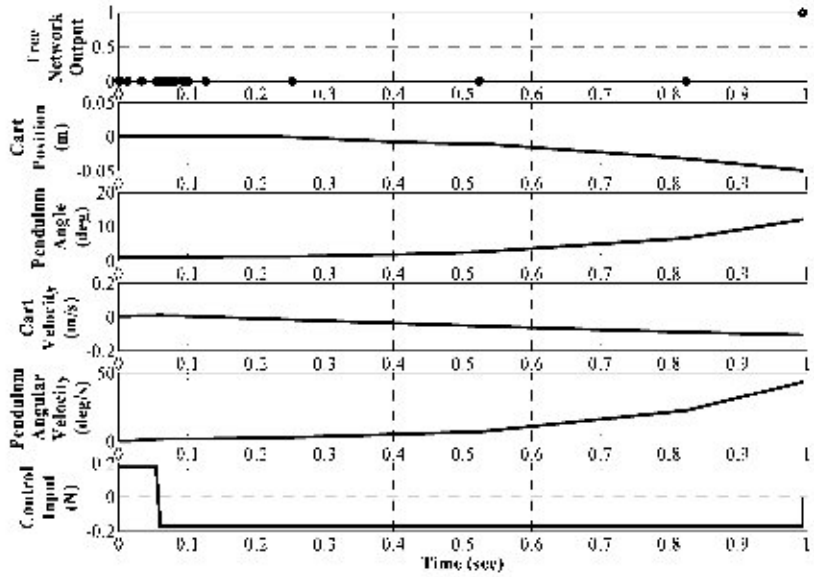

Fig. 10. Sample trajectories from the ADHDP training scheme of the cart-pole system. The failure condition occurs when the pendulum angle falls past the positive limit.

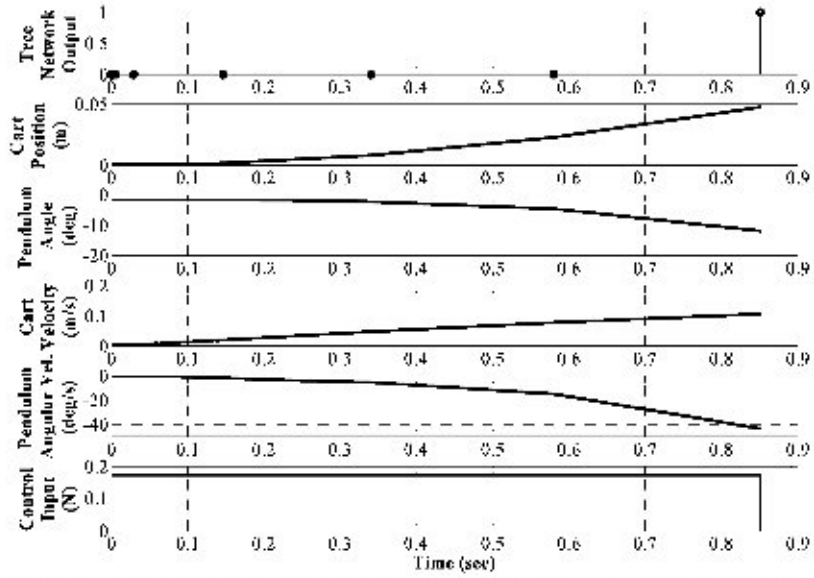

Fig. 11. Sample trajectories from the ADHDP training scheme of the cart-pole system. The failure condition occurs when the pendulum angle falls past the negative limit.

corresponding utility value act as the initial training set for the SVM-based tree-type NN.

The plant was simulated with the initialized states, and training data was collected for both the critic and action networks. Once a failed state is reached (which is indicated by the critic output of $+1$), both the critic and action elements need to be updated. Once a batch of failure data is obtained by conducting the experiments from randomly generated initial conditions over the state space, a new critic has to be trained. Once a new critic has been obtained, out of a set $\{u_i; i = 1, 2, \ldots k; u_i \in (-1, 1)\}$ of k available controls, the proper control action is chosen. This is the action that produces a zero at the output of the critic due to failure state, as well as provides a control force in the right direction to balance the cart-pole system. Once the failure states and their desired control actions to balance the pole are known, the LS-SVM regressor is trained. The regularization parameter γ and the kernel parameter σ^2 were tuned by using the function *tunelssvm* of the LS-SVM toolbox [32]. This cycle of training the critic and the action element is continued whenever the critic network indicates a failure. The larger the number of zero values at the output of the critic is, the more the value function $J(\cdot)$ is reduced. The flowchart in Fig. 8 illustrates the ADHDP-based controller training in which the SVM-based tree-type NN has been used as the "critic" and an LS-SVM regressor as the "action" element.

The trained ADHDP-based controller had a 14-node SVM-based tree-type NN as the critic (Fig. 9), and the parameters of the LS-SVM regressor-based action element were $\gamma = 94.111$ and $\sigma^2 = 5.679$.

D. Simulation Results

The sample performance of the tree network as the critic and the progress of the state trajectories of the cart-pole system

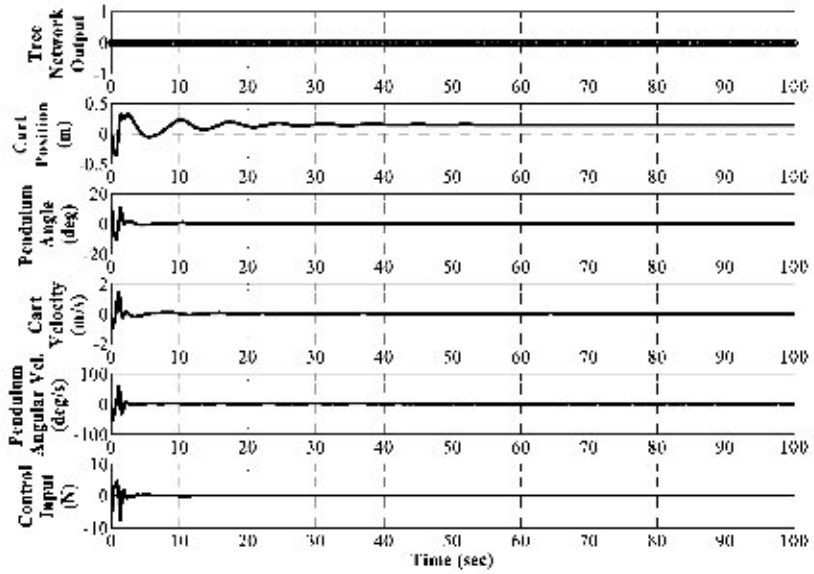

Fig. 12. Balancing the pendulum by the trained ADHDP-based controller. The initial angle of the pendulum is 10.8° from the vertical.

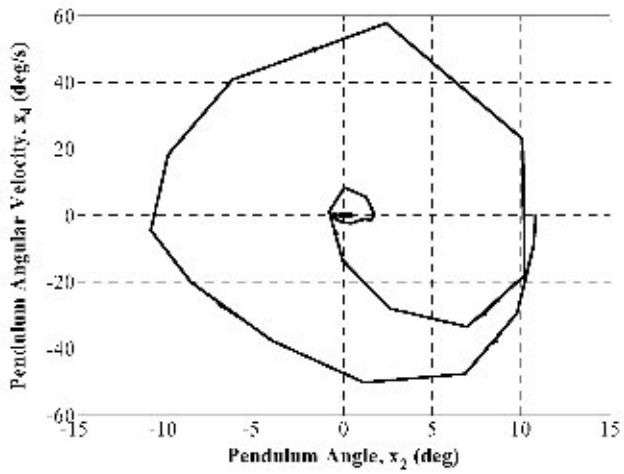

Fig. 13. Phase-plane plots showing convergence of the ADHDP-based control scheme: pendulum angular velocity (x_4) versus pendulum angle (x_2).

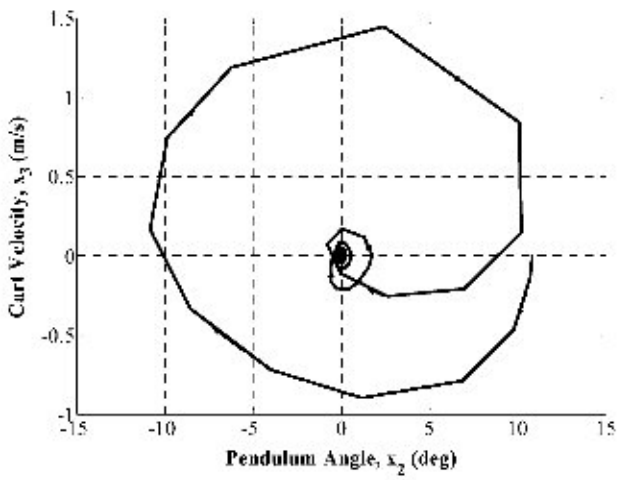

Fig. 14. Phase-plane plots showing convergence of the ADHDP-based control scheme: cart velocity (x_3) versus pendulum angle (x_2).

with the input during training of the critic and action elements in tandem have been demonstrated. As the pole angle falls past the vertical beyond $+12°/-12°$, the output of the critic switches from 0 to 1, indicating the failure condition and stopping further control input to the plant, as shown in Figs. 10 and 11.

Fig. 12 shows the performance of a trained ADHDP-based controller with the pole initially placed at some arbitrary angle (say, 10.8°) from the vertical. The controller successfully steers the plant and it achieves an equilibrium state of

$$\mathbf{x}^e = [0.146 \ 0 \ 0 \ 0]^T.$$

The safe operation of the plant has been indicated by zero outputs at the output of the tree network, while the Gaussian-kernel-based LS-SVM regressor provides smooth control.

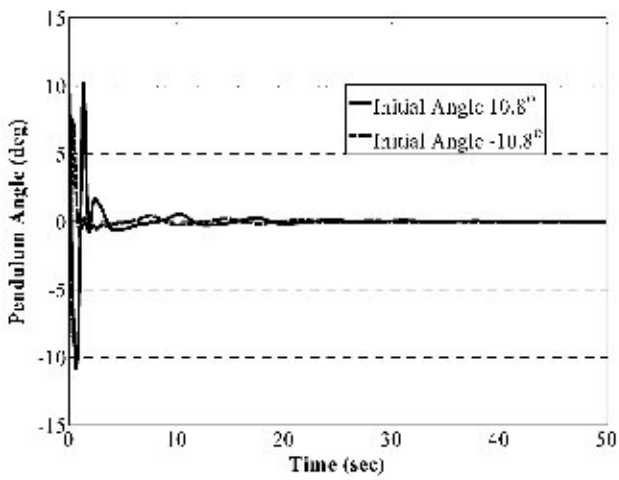

Fig. 15. Angular deviation of the pendulum starting from initial pendulum angles on both sides of the vertical at $+10.8°$.

The main objective of the pole-balancing problem is to keep the state of pole angle vertical within the bounds of the track.

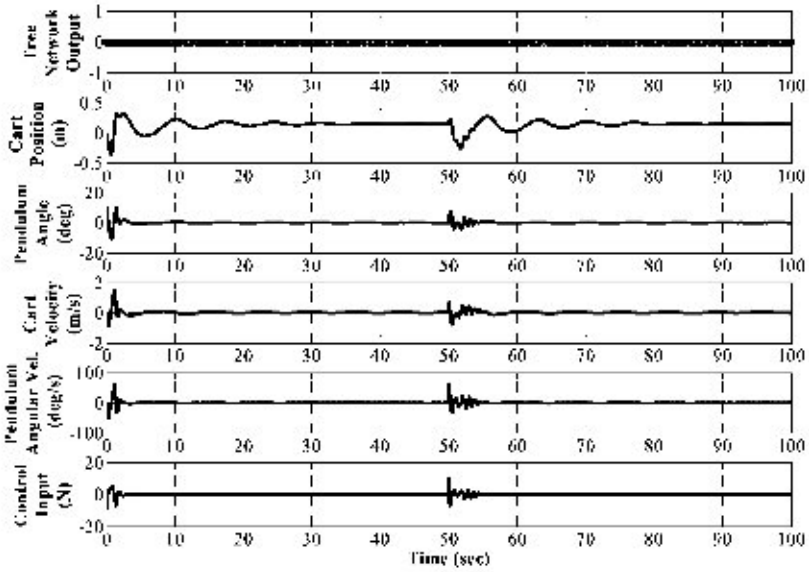

Fig. 16. Effect of disturbance due to an impulse input at the fiftieth second.

Convergence of the velocity components of an equilibrium state $\boldsymbol{x}^*$ to the origin ensures stability of the plant [33]. The simulation data for an initial pole angle of -10.8° has been used to draw the phase-plane trajectories shown in Figs. 13 and 14. The trajectories depict the convergence of the pendulum angular velocity and the cart velocity to the origin, along with the pendulum vertical angle. The coarseness of the time intervals for the control actions shows the controlling capability of the technique. With the final cart position well inside the track, the ADHDP-based controller ensures smooth operation of the cart-pole system.

The performance of the ADHDP-based control scheme for the pole-balancing problem was tested for the initial position of the pole on both sides of the vertical. Fig. 15 shows the trajectory of the pendulum angle, for initial angles of the pole at 10.8° and -10.8° from the vertical. The Gaussian-kernel-based LS-SVM regressor that acts as the action element ensures smooth control while balancing the pole.

In order to provide a comparison, we used the same initial conditions as that in Lin and Lin [21], viz. the initial cart position = 0.1 m, initial pendulum angle = 5.8°, initial cart velocity = 0.01 m/s, and initial angular velocity of the pendulum = 0.058°/s. The other parameters of the cart-pendulum setup are nearly identical. In the proposed approach, the mean deviation of the pendulum was 0.146°, as against 0.5° in [21].

E. Controller Robustness

One of the attributes desired of a controller is its ability to perform under unforeseen conditions. These conditions may be manifested in a control system as external disturbances, or changes in plant operating conditions. A controller should be capable to operate the plant even under such conditions.

1) Disturbance Rejection: To demonstrate the disturbance rejection capability of a controller trained by the ADHDP-based control scheme, an impulse input was applied on the cart at the fiftieth second when the plant states had achieved steady state. A force of 10 N was applied for 0.1 s at the fiftieth second, to generate the impulse input. Fig. 16 shows the corrective actions taken by the LS-SVM controller to quickly restore the pole back to the stable condition on application of the disturbance. Throughout the simulation period, the SVM-based tree-type NN that acts as the critic generates a zero output, indicating stable operation of the plant.

2) Change in Plant Parameters: To show the effect of changes in the plant parameters, the pole length was reduced to half of its original length. Fig. 17 shows the simulation results of controlling such a pole by the trained ADHDP-based controller. The trained controller takes prompt and smooth corrective action to restore the pole to the vertical position. In another test, to mimic the changes in plant parameters, the mass of the cart was doubled. Fig. 18 shows how the trained ADHDP-based controller provides necessary control inputs that move the cart both ways about the center of the track, to ultimately restore the pole to the vertical position.

These robustness tests show good control capabilities of the ADHDP-based controller for appreciable changes in plant parameters, and also demonstrate its disturbance rejection performance. The response to plant parameter changes demonstrates the real-time learning capability of the controller to handle unforeseen situations.

VI. Conclusion

In this paper, we have proposed applying the SVM-based tree-type NN [8] as the critic in the ADHDP strategy for control. Conventional feedforward ANNs, when used as the critic, suffer from some drawbacks in such an application. Since the failure state is rare, failure instances may be difficult to learn in the midst of a large number of exemplars of the other class [7]. The empiricism inherent in the application of ANNs has been pointed out in the literature—the number of hidden nodes, layers, and learning parameters requires a careful choice [24], and the solution may have a high sensitivity around optimal choice values. Error backpropagation-based learning algorithms also suffer from being entrapped in local minima. The use of

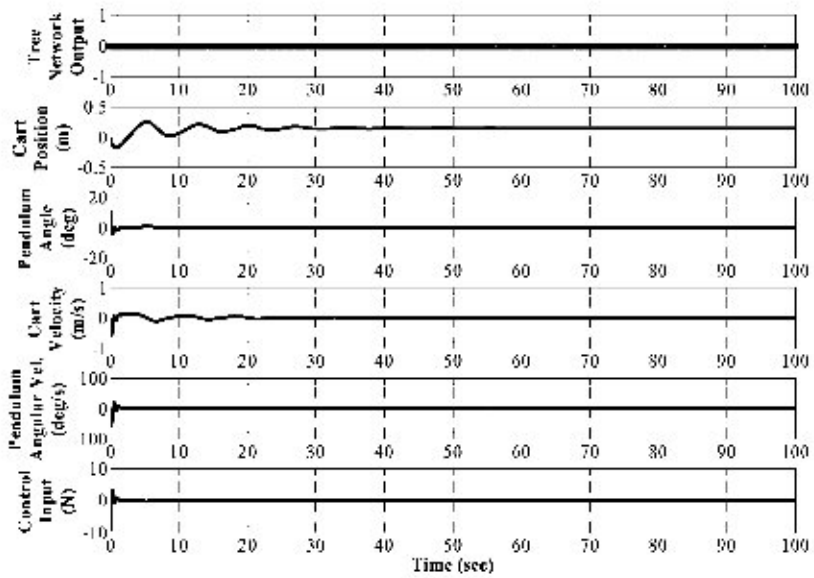

Fig. 17. Effect due to change in plant parameters when the pole length is reduced to half.

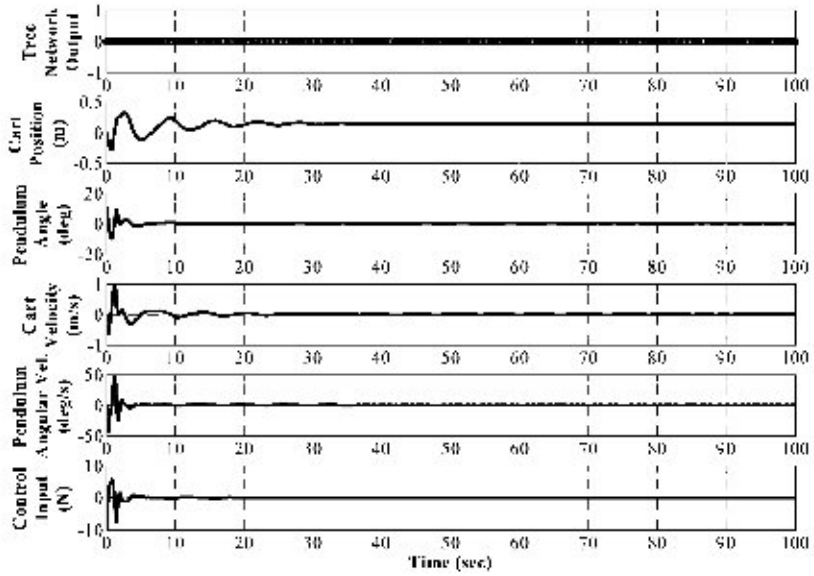

Fig. 18. Effect due to change in plant parameters when the mass of the cart is doubled.

SVM-based tree-type NN that guarantees complete learning of any data set for a binary classification task, has helped ameliorate some of these problems. This also facilitates the learning of rare events such as failure states, which is important in the ACD setting.

SVMs address the problem of maximizing generalizability while simultaneously minimizing the classification error. ANN training methods are usually based on minimizing the mean squared error over a set of training exemplars. In this paper, the critic target of learning a binary mapping has been formulated in a linear programming framework. Advantages of using linear programming are its simplicity, possibility of parallelism, and optimality of the solution. The superior function approximation capability of an LS-SVM regressor with a Gaussian kernel has been used to implement the controller. The "highly nonlinear" inverted pendulum model was chosen as the plant to implement the control strategy. The proposed type of control scheme offers an effective way of "failure avoidance control," where the LS-SVM controller provides near-optimal control for a highly nonlinear plant.

Centering the Gaussian kernels on the failure states appears to contribute to robustness. At the same time, the SVM-based tree-type NN that is capable of learning any binary classification data provides fault tolerance to the control system. The ability of the trained controller to reject disturbances and also to maneuver the plant under adverse conditions of plant parameter changes demonstrates its robust performance.

The tree may be more amenable to being interpreted in the form of a set of sequential decisions or rules. Such a tree-type network can also be trained to learn sequences, in which the present output is a function of past and present inputs, as well as past outputs [34]. This opens up the possibility of an SVM-based tree-type network being used in place of an ANN in a fully recurrent ACD–ANN setting.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers and Prof. S. C. Dutta Roy for their valuable comments and critical appraisal of the manuscript.

REFERENCES

[1] G. Cybenko, "Approximation by superposition of a sigmoidal function," *Math. Control Signals Syst.*, vol. 2, pp. 303–314, 1989.

[2] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, pp. 359–366, 1989.

[3] P. J. Werbos, "A menu of designs for reinforcement learning over time," in *Neural Networks for Control*, W. Miller, R. Sutton, and P. Werbos, Eds. Cambridge, MA: MIT Press, 1990.

[4] D. V. Prokhorov, R. A. Santiago, and D. C. Wunsch, II, "Adaptive critic designs: A case study for neurocontrol," *Neural Netw.*, vol. 8, no. 9, pp. 1367–1372, 1995.

[5] D. V. Prokhorov, "Adaptive critic designs and their applications," Ph.D. dissertation, Dept. Electr. Eng., Texas Tech Univ., Lubbock, TX, Oct. 1997.

[6] D. V. Prokhorov and D. C. Wunsch, II, "Adaptive critic designs," *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 997–1007, Sep. 1997.

[7] D. Liu, "Adaptive critic designs for problems with known analytical form of cost function," in *Proc. Int. Joint Conf. Neural Netw.*, Honolulu, HI, May 12–17, 2002, vol. 3, pp. 1808–1813.

[8] A. K. Deb, Jayadeva, S. Chandra, and M. Gopal, "Modified growth network for pattern classification," in *Operations Research and Its Applications: Recent Trends (APORS 2003)*, M. R. Rao and M. C. Puri, Eds. New Delhi, India: Allied Publishers, vol. I, pp. 270–277.

[9] V. Vapnik, *The Nature of Statistical Learning Theory*, 2nd ed. New York: Springer-Verlag, 2000.

[10] V. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

[11] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel Based Learning Methods*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[12] C. Saunders, A. Gammerman, and V. Vovk, "Ridge regression learning algorithm in dual variables," in *Proc. 15th Int. Conf. Mach. Learn.*, 1998, pp. 515–521.

[13] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machines classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, 1999.

[14] Jayadeva, A. K. Deb, and S. Chandra, "Binary classification by SVM-based tree type neural networks," in *Proc. Int. Joint Conf. Neural Networks*, Honolulu, HI, May 12–17, 2002, vol. 3, pp. 2773–2778.

[15] G. Martinelli, L. P. Ricotti, S. Ragazzini, and F. M. Mascioli, "A pyramidal delayed perceptron," *IEEE Trans. Circuits Syst.*, vol. 37, no. 9, pp. 1176–1181, Sep. 1990.

[16] J. A. K. Suykens, J. Vandewalle, and B. De. Moor, "Optimal control by least squares support vector machines," *Neural Netw.*, vol. 14, pp. 23–35, 2001.

[17] Feedback Instruments Limited, "Feedback MATLAB based control products—Digital pendulum control systems," Crowborough, E. Sussex, UK.

[18] V. Biega and S. N. Balakrishnan, "A dual neural network architecture for linear and nonlinear control of inverted pendulum on a cart," in *Proc. IEEE Int. Conf. Control Appl.*, 1996, pp. 614–619.

[19] G. K. Venayagamoorthy, R. G. Harley, and D. C. Wunsch, II, "Implementation of adaptive critic-based neurocontroller for turbogenerators in multimachine power systems," *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1047–1064, Sep. 2003.

[20] H. R. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 724–740, Sep. 1992.

[21] C. J. Lin and C. T. Lin, "Reinforcement learning for an ART-based fuzzy adaptive learning control network," *IEEE Trans. Neural Netw.*, vol. 7, no. 3, pp. 709–731, May 1996.

[22] J. M. Zurada, *Introduction to Artificial Neural Systems*. Mumbai, India: Jaico Publishing House, 1997.

[23] N. K. Bose and P. Liang, *Neural Network Fundamentals with Graphs, Algorithms and Applications*. New Delhi, India: Tata/McGraw-Hill, 1998.

[24] Jayadeva, A. K. Deb, and S. Chandra, "Algorithm for building a neural network for function approximation," *Inst. Electr. Eng. Proc.—Circuits, Devices, Syst.*, vol. 149, no. 5/6, pp. 301–307, 2002.

[25] J. A. Bullinaria, "Evolving efficient learning algorithms for binary mappings," *Neural Netw.*, vol. 16, pp. 793–800, 2003.

[26] O. L. Mangasarian, "Mathematical programming in neural networks," *ORSA J. Comput.*, vol. 5, no. 4, pp. 349–360, 1993.

[27] K. P. Bennett and O. L. Mangasarian, "Multicategory discrimination via linear programming," *Optim. Methods Software*, vol. 3, pp. 27–39, 1994.

[28] A. Roy, S. Govil, and R. Miranda, "An algorithm to generate Radial Basis Function (RBF)-like nets for classification problems," *Neural Netw.*, vol. 8, no. 2, pp. 179–201, 1995.

[29] A. Roy, S. Govil, and R. Miranda, "A neural-network learning theory and a polynomial time RBF algorithm," *IEEE Trans. Neural Netw.*, vol. 8, no. 6, pp. 1301–1313, Nov. 1997.

[30] A. Roy and S. Mukhopadhyay, "Iterative generation of higher-order nets in polynomial time using linear programming," *IEEE Trans. Neural Netw.*, vol. 8, no. 2, pp. 402–412, Mar. 1997.

[31] B. L. Lu, H. Kita, and Y. Nishikawa, "Inverting feedforward neural networks using linear and nonlinear programming," *IEEE Trans. Neural Netw.*, vol. 10, no. 6, pp. 1271–1290, Nov. 1990.

[32] K. Pelckmans, J. A. Suykens, T. V. Gestel, J. D. Brabanter, L. Lukas, B. Hamars, B. D. Moor, and J. Vandewalle, LS-SVMlab Toolbox User's Guide, [Online]. Available: http://www.esat.kuleuven.ac.be/sista/lssvmlab/

[33] M. Gopal, "Lyapunov stability analysis," in *Digital Control and State Variable Methods*, 2nd ed. New Delhi, India: Tata/McGraw, 2003, ch. 8.

[34] Jayadeva and A. K. Deb, "Learning sequences using tree type neural networks," in *VI Int. Conf. Cogn. Neural Syst.*, Boston, MA, May 29–Jun. 1 2002, p. 90.

Alok Kanti Deb (S'02) was born in Ranaghat, West Bengal, India. He received the B.E. degree with first class from the Bengal Engineering College, Calcutta University, Shibpur, India and the M. Tech and Ph.D. degrees from the Indian Institute of Technology (IIT), New Delhi, India, in 1994, 1999, and 2006, respectively, all in electrical engineering.

Currently, he is an Assistant Professor at the Center for Soft Computing Research: A National Facility, Indian Statistical Institute, Kolkata, India. His research interests are in computational intelligence and control systems.

Dr. Deb received the student travel award from the IEEE Neural Network Society for attending the IEEE World Congress on Computational Intelligence (WCCI 2002). He was a Program Committee member of FUZZ-IEEE 2005, Reno, NV.

Jayadeva (M'94–SM'04) received the B.Tech. and Ph.D. degrees from the Indian Institute of Technology (IIT), New Delhi, India, in 1988 and 1993, respectively.

Currently, he is a Professor at the Department of Electrical Engineering, IIT Delhi. He was a Young Scientist of the International Union of Radio Science in 1996. He visited the Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, in summer 1997 as a Better Opportunities for Young Scientists in Chosen Areas of Sciences and Technology (BOYSCAST) Fellow from the Department of Science and Technology, Government of India. His current research interests include machine learning, biological and artificial neural systems, optimization, and very large scale integration (VLSI) design and computer-aided design (CAD).

Dr. Jayadeva is the recipient of the Indian National Science Academy's Medal for Young Scientists, the Indian National Academy of Engineering's Young Engineer Award, and the IEEE Aerospace and Communication (AESCOM) Chapter India Award in the area of electronic systems. He is an Honorary Editor of the Institution of Electronics and Telecommunication Engineers, India (IETE) *Journal of Research*. He is a speaker from India on the IEEE Computer Society Distinguished Visitor Program.

Madan Gopal received the B.Tech. degree in electrical engineering, the M.Tech. degree in control systems, and the Ph.D. degree in control systems from Birla Institute of Technology, Pilani, India, in 1968, 1970, and 1976, respectively.

Currently, he is a Professor at the Department of Electrical Engineering, Indian Institute of Technology (IIT), New Delhi, India. His teaching and research stints span over three decades at prestigious institutes. He is the author/coauthor of six books. His video course on control engineering is being transmitted periodically through EKLAVYA Technology Channel of IITs. He is author of an interactive web compatible multimedia course on control engineering. He has a large number of research publications to his credit. His current research interests are in the areas of machine learning, soft-computing technologies, and intelligent control.

Suresh Chandra received the M.S. degree in mathematical statistics from Lucknow University, India, in 1965 and the Ph.D. degree in mathematics (mathematical programming) from the Indian Institute of Technology (IIT), Kanpur, India, in 1970.

Currently, he is a Professor at the Department of Mathematics, Indian Institute of Technology, Delhi, India. He has authored and coauthored more than 100 publications in refereed journals and international conferences and coauthored two books, one on fuzzy mathematical programming and fuzzy matrix games and the other on principles of optimization theory. His research interests include numerical optimization, mathematical programming, generalized convexity, fuzzy optimization, fuzzy games, NNs, machine learning, and financial mathematics.

Dr. Chandra is a Member of the Editorial Board for the *International Journal of Management and Systems* and the *Journal of Decision Sciences*. He is also a Senior Member of the Operational Research Society of India and a Member of the International Working Group on Generalized Convexity and Applications.