Chapter 8

LS-SVM for Recurrent Networks and Control

The problems discussed in the previous Chapters were either supervised or unsupervised but on the other hand always static in the sense that there are no recursive equations involved in these formulations. In this Chapter we show how LS-SVM methods can be extended to problems with dynamics, more specifically an extension is made from feedforward to recurrent LS-SVMs and the use of LS-SVMs in the context of optimal control. Although the resulting problems are non-convex one can still apply the kernel trick after formulating the primal problem and taking conditions for optimality from the Lagrangian.

8.1 Recurrent Least Squares Support Vector Machines

8.1.1 *From feedforward to recurrent LS-SVM formulations*

The methods for nonlinear function estimation by LS-SVM as discussed in previous Chapters can be used in a dynamical systems context only in combination with NARX models

$$\hat{y}_k = f(y_{k-1}, y_{k-2}, ..., y_{k-p}, u_{k-1}, u_{k-2}, ..., u_{k-p}) \tag{8.1}$$

where $\hat{y}$ denotes the estimated output, y the true output and f a smooth nonlinear mapping. This model structure is intended for modelling a dynamical system with input $u \in \mathbb{R}$ and output $y \in \mathbb{R}$ with discrete time index k. The value p denotes the model order. On the other hand, the model is essentially feedforward despite the fact that it is used to model a dynamical system. When using static LS-SVM models the given training data set con-

sists of given input data vectors $\{[y_{k-1}; y_{k-2}; ...; y_{k-p}; u_{k-1}; u_{k-2}; ...; u_{k-p}]\}_{k=p+1}^{p+N}$ and corresponding given target output values $\{y_k\}_{k=p+1}^{p+N}$. Although this model structure is employed to model dynamics of a system, it is inherently feedforward due to the fact that the equation is not recursive.

A *recurrent model* [230; 231] on the other hand in input-output description takes the form

$$\hat{y}_k = f(\hat{y}_{k-1}, \hat{y}_{k-2}, ..., \hat{y}_{k-p}, u_{k-1}, u_{k-2}, ..., u_{k-p}) \tag{8.2}$$

which is a recursive equation in the estimated output. This is a nonlinear output error model (NOE). Let us further study now the autonomous case of this recurrent model

$$\hat{y}_k = f(\hat{y}_{k-1}, \hat{y}_{k-2}, ..., \hat{y}_{k-p}) \tag{8.3}$$

and explain how LS-SVM methodology can be used in the context of such model structures. We will do this again by stating the problem in the primal weight space as a constrained optimization problem. Next we construct the Lagrangian, take conditions for optimality and solve the dual problem in the Lagrange multipliers.

The following model and optimization problem were formulated by Suykens & Vandewalle in [240]. The recurrent model in the primal weight space is given by

$$\hat{y}_k = w^T \varphi(\begin{bmatrix} \hat{y}_{k-1} \\ \hat{y}_{k-2} \\ \vdots \\ \hat{y}_{k-p} \end{bmatrix}) + b, \tag{8.4}$$

where $\varphi(\cdot)$ is the mapping to the high dimensional feature space. The initial condition for this model is assumed to be given. Starting from the initial condition one can simulate the system (8.4) in time to generate a sequence of values $\{\hat{y}\}$, while the model (8.1) is simply a one step ahead predictor model. The model (8.4) is expressed then in terms of the true measured output values and a set of unknown error variables:

$$y_k - e_k = w^T \varphi(y_{k-1\_k-p} - e_{k-1\_k-p}) + b \tag{8.5}$$

where $e_k = y_k - \hat{y}_k$, $y_{k-1\_k-p} = [y_{k-1}; y_{k-2}; ...; y_{k-p}]$, $e_{k-1\_k-p} = [e_{k-1}; e_{k-2}; ...; e_{k-p}]$ by definition. The output weight vector and bias term are denoted by $w \in \mathbb{R}^{n_h}$ and $b \in \mathbb{R}$. Note that recurrent neural networks are clas-

sically trained by dynamic backpropagation or backpropagation through time [170; 289], which is more complicated than a backpropagation method for training of feedforward neural networks.

The recurrent LS-SVM approach starts from the following constrained optimization problem in primal weight space [240]

$$\left[\begin{array}{l} \boxed{\mathrm{P}}: \quad \min_{w,b,e} J_{\mathrm{P}}(w,e) = \frac{1}{2}w^T w + \gamma\frac{1}{2}\sum_{k=p+1}^{p+N} e_k^2 \\ \qquad \text{such that} \qquad y_k - e_k = w^T\varphi(y_{k-1\_k-p} - e_{k-1\_k-p}) + b, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad k = 1,...,N. \end{array}\right] \tag{8.6}$$

At this point it is important to note that the constraints are *nonlinear* in the unknown e variables, while in the static LS-SVM case the constraints are linear in the unknowns w, b, e. Hence, the problem will become non-convex in general with a non-unique solution. The Lagrangian for the problem is

$$\mathcal{L}(w,b,e;\alpha) = \mathcal{J}(w,b,e) + \sum_{k=p+1}^{p+N} \alpha_{k-p}\,[y_k - e_k - w^T\,\varphi(y_{k-1\_k-p} - e_{k-1\_k-p}) - b]. \tag{8.7}$$

The conditions for optimality are given by

$$\left\{\begin{array}{rcl} \frac{\partial\mathcal{L}}{\partial w} &=& w - \sum_{k=p+1}^{p+N} \alpha_{k-p}\,\varphi(y_{k-1\_k-p} - e_{k-1\_k-p}) = 0 \\ \frac{\partial\mathcal{L}}{\partial b} &=& \sum_{k=p+1}^{p+N} \alpha_{k-p} = 0 \\ \frac{\partial\mathcal{L}}{\partial e_k} &=& \gamma e_k - \alpha_{k-p} - \sum_{i=1}^{p} \alpha_{k-p+i}\frac{\partial}{\partial e_{k-i}}[w^T\varphi(y_{k-1\_k-p} \\ && -e_{k-1\_k-p})] = 0, \qquad\qquad k = p+1,...,N \\ \frac{\partial\mathcal{L}}{\partial\alpha_{k-p}} &=& y_k - e_k - w^T\,\varphi(y_{k-1\_k-p} - e_{k-1\_k-p}) - b = 0, \\ && \qquad\qquad k = p+1,...,p+N. \end{array}\right. \tag{8.8}$$

It is difficult to eliminate the unknowns e_k from this set of nonlinear equations. However, it is straightforward to eliminate w (which can be infinite dimensional) and apply the kernel trick by $K(y_{k-1\_k-p}-e_{k-1\_k-p}, y_{l-1\_l-p}-e_{l-1\_l-p}) = \varphi(y_{k-1\_k-p}-e_{k-1\_k-p})^T\varphi(y_{l-1\_l-p}-e_{l-1\_l-p})$.

This leads to the following set of nonlinear equations:

PD : solve set of nonlinear equations in α, e, b :

$$\begin{cases} \sum_{k=p+1}^{p+N} \alpha_{k-p} = 0 \\ \gamma e_k - \alpha_{k-p} - \sum_{i=1}^{p} \alpha_{k-p+i} \frac{\partial}{\partial e_{k-i}} \Big(\sum_{l=p+1}^{p+N} \alpha_{l-p} K(y_{k-1\_k-p} - \\ \quad e_{k-1\_k-p}, y_{l-1\_l-p} - e_{l-1\_l-p}) = 0, \qquad k = p+1, ..., N \\ y_k - e_k - \sum_{l=p+1}^{p+N} \alpha_{l-p} K(y_{k-1\_k-p} - e_{k-1\_k-p}, y_{l-1\_l-p} \\ \quad -e_{l-1\_l-p}) - b = 0, \qquad k = p+1, ..., p+N. \end{cases} \tag{8.9}$$

8.1.2 *A simplification to the problem*

Finding a solution to the equations (8.9) is computationally heavy. A simplification can be made by letting the regularization constant $\gamma \to \infty$. This leads to the problem:

PD : $\min_{e,b,\alpha} J_{\mathrm{PD}}(e) = \frac{1}{2} \sum_{k=p+1}^{p+N} e_k^2$

such that

$$\begin{aligned} y_k - e_k = & \sum_{l=p+1}^{p+N} \alpha_{l-p} K(y_{l-1\_l-p} - \\ & e_{l-1\_l-p}, y_{k-1\_k-p} - e_{k-1\_k-p}) + b, \qquad k = p+1, ..., N+p \end{aligned}$$

$$\sum_{k=p+1}^{p+N} \alpha_{k-p} = 0. \tag{8.10}$$

This constrained nonlinear optimization problem can be solved e.g. by sequential quadratic programming (SQP) [82]. For large data sets special methods for large scale nonlinear optimization exist and can be applied [174]. Because the results are based on the limit case $\gamma \to \infty$ the regularization term $\frac{1}{2}w^T w$ is partially neglected (however the form of the solution still takes into account the regularization term) there is a danger for overfitting by minimizing the training set error. Therefore, instead of minimizing the cost function to its local minimum one may apply early stopping then, which is another form of regularization. Note that standard VC bounds on the generalization error cannot be applied here due to the fact that the data are not i.i.d. because the system is dynamic instead of static.

Instead of solving this problem in the unknowns e, b, α one may further eliminate e and solve

$$\boxed{\text{D}}: \quad \min_{\alpha,b} J_{\text{D}}(\alpha, b) = \frac{1}{2} \sum_{k=p+1}^{p+N} (y_k - \hat{y}_k(\alpha))^2$$

such that

$$\hat{y}_k = \sum_{l=p+1}^{p+N} \alpha_{l-p} K(\hat{y}_{l-1\_l-p}, \hat{y}_{k-1\_k-p}) + b, \quad k = p+1, ..., N+p$$

$$\sum_{k=p+1}^{p+N} \alpha_{k-p} = 0, \tag{8.11}$$

where the recurrent simulation model can be used

$$\hat{y}_k = \sum_{l=p+1}^{p+N} \alpha_{l-p} K(\hat{y}_{l-1\_l-p}, \begin{bmatrix} \hat{y}_{k-1} \\ \hat{y}_{k-2} \\ \vdots \\ \hat{y}_{k-p} \end{bmatrix}) + b \tag{8.12}$$

with given initial condition $\hat{y}_i = y_i$ for $i = 1, 2, ..., p$. For RBF kernels one employs

$$K(\hat{y}_{k-1\_k-p}, \hat{y}_{l-1\_l-p}) = \exp(-\nu \|\hat{y}_{k-1\_k-p} - \hat{y}_{l-1\_l-p}\|_2^2) \tag{8.13}$$

where ν is a positive real constant. This simplified case is equivalent to parameterizing the recurrent model structure (8.2) by a radial basis function

network and minimizing the fitting error and requiring that the average error on the training data is zero.

8.1.3 *Example: trajectory learning of chaotic systems*

We illustrate the training of recurrent LS-SVMs now on an example of trajectory learning of chaotic systems, according to [240]. A well-known paradigm for chaos in electrical circuits is Chua's circuit [43]. It is described by the following equations:

$$\begin{cases} \dot{x}_1 &= a\,[x_2 - g(x_1)] \\ \dot{x}_2 &= x_1 - x_2 + x_3 \\ \dot{x}_3 &= -bx_2 \end{cases} \tag{8.14}$$

with piecewise linear characteristic

$$g(x_1) = m_1 x_1 + \frac{1}{2}(m_0 - m_1)\,(|x_1 + 1| - |x_1 - 1|). \tag{8.15}$$

A double scroll strange attractor is obtained by taking $a = 9$, $b = 14.286$, $m_0 = -1/7$, $m_1 = 2/7$. A trajectory has been generated with initial condition $[0.1; 0; -0.1]$ for the state vector with three state variables $[x_1; x_2; x_3]$ by using a Runge-Kutta integration rule (ode23 in Matlab).

In Fig. 8.1 the first $N = 300$ data points were used for trajectory learning of the double scroll by a recurrent LS-SVM with RBF kernel. For the model structure $p = 12$ has been taken and $\nu = 1$ for the RBF kernel. In order to solve the constrained nonlinear optimization problem (8.11), sequential quadratic programming (SQP) has been applied (*constr* in Matlab with a specification of the number of equality constraints). The model (8.12) has been simulated in C by using Matlab's cmex facility. In all simulations, for the initial unknown parameter vector α_k, e_k in (8.10) have been chosen randomly according to a Gaussian distribution with zero mean and standard deviation 0.1 and $b = 0$. Although no further optimization of the model structure has been done, it has been observed that small values of p (that are chosen in accordance with Takens' embedding theorem) are slowing down the training process. In Fig. 8.1 the result after training of the recurrent LS-SVM is shown in dashed line. The model is simulated for initial condition $\hat{y}_{1\_12} = y_{1\_12}$. As one can see the training data are well reproduced 300 points ahead starting from this initial condition. The prediction beyond the time horizon of 300 training data points is very good, given the limited

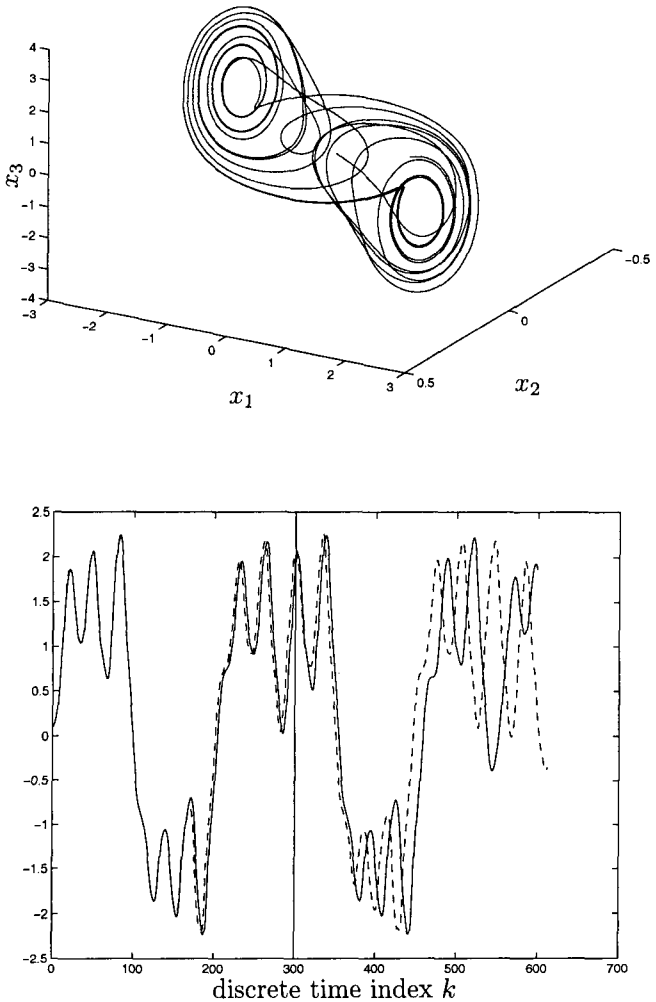

Fig. 8.1 *(Top) Double-scroll attractor; (Bottom) Trajectory learning of the double scroll (full line) by a recurrent LS-SVM with RBF kernel. The simulation result after training (dashed line) on $N = 300$ data points is shown with data points $k = 1$ to 12 i.e. $\hat{y}_{1\_12} = y_{1\_12}$ taken as initial condition. Good qualitative generalization is obtained for the prediction from $k = 301$ to about $k = 500$.*

amount of training data. The two next jumps to the other scroll are quite well predicted. The early stopping in the training procedure was done based upon the training data itself (instead of on an independent validation set). This is because it is non-trivial to reproduce the trajectory in recurrent mode on the training data itself. The overfitting was observed in recurrent mode on these training data. At that point training was stopped. The resulting recurrent LS-SVM model is generalizing well to a double scroll attractor, also for small perturbations on the initial state. One should also note that for the recurrent LS-SVMs it is important to simulate the form (8.12) because when solving the constrained nonlinear optimization problem (8.11) the constraints will usually not hold exactly but only for a certain tolerance. For chaotic systems such small perturbations could cause significant differences between the solution vector from (8.11) and the recurrent simulation model (8.12), while for globally asymptotically stable systems such perturbations are not amplified.

8.2 LS-SVMs and optimal control

Besides applying LS-SVM methods to recurrent models one can also follow a similar methodology towards solving optimal control problems. Both the optimal control problems and support vector methods are closely related to optimization theory. Hence one could try to merge the two formulations. This approach has been originally proposed by Suykens *et al.* in [241]. Let us first discuss a few aspects of the N-stage optimal control problem, next explain how one embeds LS-SVMs in the problem formulation and finally show how to apply the kernel trick and solve the problem in the Lagrange multipliers. In control problems stability and robustness are important issues. These aspects are discussed in the next subsection together with illustrative examples.

8.2.1 *The N-stage optimal control problem*

In the N-stage optimal control problem one aims at solving the following problem [33]:

$$\min \mathcal{J}_N(x_k, u_k) = \rho(x_{N+1}) + \sum_{k=1}^{N} h(x_k, u_k) \tag{8.16}$$

subject to the system dynamics

$$x_{k+1} = f(x_k, u_k), \quad k = 1, ..., N \quad (x_1 \text{ given}),$$

where $\rho(\cdot)$ and $h(\cdot, \cdot)$ are positive definite functions and N is the finite discrete time horizon. A typical choice is the quadratic cost $h(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k$, $\rho(x_{N+1}) = x_{N+1}^T Q x_{N+1}$ with $Q = Q^T > 0$, $R = R^T > 0$. The functions $\rho(\cdot)$, $h(\cdot, \cdot)$, $f(\cdot, \cdot)$ are assumed to be twice continuously differentiable, $x_k \in \mathbb{R}^n$ denotes the state vector, $u_k \in \mathbb{R}$ is the input of the system with discrete time index k. We consider here a single input system but it can be extended to multi-input systems as well.

In order to find the optimal control law, one constructs the Lagrangian

$$\mathcal{L}_N(x_k, u_k; \lambda_k) = \mathcal{J}_N(x_k, u_k) + \sum_{k=1}^{N} \lambda_k^T [x_{k+1} - f(x_k, u_k)] \tag{8.17}$$

with Lagrange multipliers $\lambda_k \in \mathbb{R}^n$. The conditions for optimality are

$$\begin{cases} \frac{\partial \mathcal{L}_N}{\partial x_k} = \frac{\partial h}{\partial x_k} + \lambda_{k-1} - (\frac{\partial f}{\partial x_k})^T \lambda_k = 0, \quad k = 2, ..., N & \text{(adjoint equation)} \\ \frac{\partial \mathcal{L}_N}{\partial x_{N+1}} = \frac{\partial \rho}{\partial x_{N+1}} + \lambda_N = 0 & \text{(adjoint final condition)} \\ \frac{\partial \mathcal{L}_N}{\partial u_k} = \frac{\partial h}{\partial u_k} - \lambda_k^T \frac{\partial f}{\partial u_k} = 0, \quad k = 1, ..., N & \text{(variational condition)} \\ \frac{\partial \mathcal{L}_N}{\partial \lambda_k} = x_{k+1} - f(x_k, u_k) = 0, \quad k = 1, ..., N & \text{(system dynamics).} \end{cases} \tag{8.18}$$

For the case of a quadratic cost function subject to linear system dynamics with infinite time horizon, the optimal control law can be represented by full static state feedback control (which is the well-known LQR (Linear Quadratic Regulator) problem). However, in general, the optimal control law cannot be represented by state feedback as the optimal control law may also depend on the co-state λ_k. Nevertheless, one is often interested in finding a suboptimal control strategy of this form for example in state feedback form $u_k = g(x_k)$. The nonlinear function $g(\cdot)$ can be parameterized by a linear function resulting in a linear state feedback controller. An alternative which has been studied in the area of neural networks is to parameterize it by means of a multilayer perceptron, which results in a powerful non-

linear state feedback law [230]. It has been shown for example in [230; 232] how difficult control problems such as swinging up an inverted and double inverted pendulum for a cart-pole system with local stabilization at the endpoint can be performed well by simple MLP neural networks.

For MLPs it is straightforward to parameterize the control law by the architecture. However, when one intends to do the same using SVMs, this is more complicated due to the nonparametric nature.

8.2.2 *LS-SVMs as controllers within the N-stage optimal control problem*

Formulation with error variables

We discuss now the approach outlined in [241], for introducing LS-SVMs within the N-stage optimal control problem formulation. The fact that LS-SVMs work with equality constraints and sum squared error simplifies the formulation in comparison with standard SVMs. One starts again by formulating the primal problem, then constructs the Lagrangian and takes the conditions for optimality. The equations for the Lagrange multipliers related to the LS-SVM control law and the Lagrange multipliers related to the system dynamics will become coupled.

The problem formulation starts from

$$\left[\begin{array}{l} \boxed{\mathrm{P}}: \quad \min \quad J_{\mathrm{P}}(x_k, u_k, w, e_k) = \mathcal{J}_N(x_k, u_k) + \frac{1}{2} w^T w + \gamma \frac{1}{2} \sum_{k=1}^{N} e_k^2 \\ \quad \text{subject to the system dynamics} \\ \qquad x_{k+1} = f(x_k, u_k), \quad k = 1, ..., N \quad (x_1 \text{ given}) \\ \quad \text{and the control law} \\ \qquad u_k = w^T \varphi(x_k) + e_k, \quad k = 1, ..., N \end{array}\right] \tag{8.19}$$

where $\mathcal{J}_N$ is given by (8.16), $\varphi(\cdot) : \mathbb{R}^n \to \mathbb{R}^{n_h}$ with n_h the dimension of the high dimensional feature space, which can be infinite dimensional. The control law is restricted to a nonlinear state feedback which is assumed to be representable by a LS-SVM. A promising aspect of using SVM here is that they are suitable for learning and generalization in large dimensional input

spaces (in this case the state space), while other methods of fuzzy control or control by RBF neural networks suffer from the curse of dimensionality as explained e.g. by Sanner & Slotine in [197]. The actual control signal applied to the plant is assumed to be $w^T\varphi(x_k)$. In the linear control case one has $\varphi(x_k) = x_k$ with $n_h = n$. In the sequel we will employ RBF kernels.

The Lagrangian takes the form

$$\mathcal{L}(x_k, u_k, w, e_k; \lambda_k, \alpha_k) = \mathcal{J}_N(x_k, u_k) + \frac{1}{2}w^T w + \gamma\frac{1}{2}\sum_{k=1}^{N} e_k^2 + \sum_{k=1}^{N}\lambda_k^T[x_{k+1} - f(x_k, u_k)] + \sum_{k=1}^{N}\alpha_k[u_k - w^T\varphi(x_k) - e_k]. \tag{8.20}$$

The conditions for optimality are given by

$$\begin{cases} \frac{\partial\mathcal{L}}{\partial x_k} = \frac{\partial h}{\partial x_k} + \lambda_{k-1} - (\frac{\partial f}{\partial x_k})^T\lambda_k - \alpha_k\frac{\partial}{\partial x_k}[w^T\varphi(x_k)] = 0, & k = 2, ..., N \quad \text{(adjoint equation)} \\ \frac{\partial\mathcal{L}}{\partial x_{N+1}} = \frac{\partial\rho}{\partial x_{N+1}} + \lambda_N = 0 \quad \text{(adjoint final condition)} \\ \frac{\partial\mathcal{L}}{\partial u_k} = \frac{\partial h}{\partial u_k} - \lambda_k^T\frac{\partial f}{\partial u_k} + \alpha_k = 0, & k = 1, ..., N \quad \text{(variational condition)} \\ \frac{\partial\mathcal{L}}{\partial w} = w - \sum_{k=1}^{N}\alpha_k\varphi(x_k) = 0 \quad \text{(support vectors)} \\ \frac{\partial\mathcal{L}}{\partial e_k} = \gamma e_k - \alpha_k = 0 & k = 1, ..., N \quad \text{(support values)} \\ \frac{\partial\mathcal{L}}{\partial\lambda_k} = x_{k+1} - f(x_k, u_k) = 0, & k = 1, ..., N \quad \text{(system dynamics)} \\ \frac{\partial\mathcal{L}}{\partial\alpha_k} = u_k - w^T\varphi(x_k) - e_k = 0, & k = 1, ..., N \quad \text{(SVM control).} \end{cases} \tag{8.21}$$

From the variational condition one can see that the Lagrange multipliers which are related to the system dynamics interact with the support values of the LS-SVM controller. The obtained set of nonlinear equations is of the

form

$$F_1(x_k, x_{N+1}, u_k, w, e_k, \lambda_k, \alpha_k) = 0 \tag{8.22}$$

for $k = 1, ..., N$ with x_1 given. Although the resulting problem is non-convex one can still apply the kernel trick and eliminate the unknown w vector. After elimination of w, a set of nonlinear equations of the form

$$F_2(x_k, x_{N+1}, u_k, \lambda_k, \alpha_k) = 0 \tag{8.23}$$

for $k = 1, ..., N$ with x_1 given is obtained. The solution is characterized as follows:

$$\left[\begin{array}{l} \boxed{\text{PD}}: \text{ solve set of nonlinear equations in } x_k, x_{N+1}, u_k, \lambda_k, \alpha_k: \\ \left\{\begin{array}{ll} \dfrac{\partial h}{\partial x_k} + \lambda_{k-1} - (\dfrac{\partial f}{\partial x_k})^T \lambda_k - \alpha_k \displaystyle\sum_{l=1}^{N} \alpha_l \frac{\partial K(x_k, x_l)}{\partial x_k} = 0, & \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad k = 2, ..., N & \\ \dfrac{\partial \rho}{\partial x_{N+1}} + \lambda_N = 0 & \\ \dfrac{\partial h}{\partial u_k} - \lambda_k^T \dfrac{\partial f}{\partial u_k} + \alpha_k = 0, & k = 1, ..., N \\ x_{k+1} - f(x_k, u_k) = 0, & k = 1, ..., N \\ u_k - \displaystyle\sum_{l=1}^{N} \alpha_l K(x_l, x_k) - \alpha_k/\gamma = 0, & k = 1, ..., N. \end{array}\right. \end{array}\right] \tag{8.24}$$

The actual control signal applied to the plant becomes

$$u(x) = \sum_{l=1}^{N} \alpha_l K(x_l, x) \tag{8.25}$$

where $\{x_l\}_{l=1}^N, \{\alpha_l\}_{l=1}^N$ are obtained from solving the set of nonlinear equations and x_k is the actual state vector at discrete time k. The data $\{x_l\}_{l=1}^N$ are used as support vector data for the control signal and are the solution to the set of nonlinear equations, while in problems of static nonlinear func-

tion estimation the support vectors correspond to given training data.

Formulation without error variables

In the previous formulation, error variables were taken in order to allow a difference between the optimal control signal u_k and the output of the LS-SVM. It is possible to simplify this and leave out the variables e_k from the problem formulation. As a result the problem has less unknowns.

According to [241] the simplified problem formulation is

$$\min_{x_k,w} \mathcal{J}_N(x_k, u_k(w, x_k)) + \frac{1}{2} w^T w \tag{8.26}$$

subject to

$$\begin{cases} x_{k+1} = f(x_k, u_k(w, x_k)), & k = 1, ..., N \quad (x_1 \text{ given}) \\ u_k = w^T \varphi(x_k) & k = 1, ..., N. \end{cases}$$

This gives the Lagrangian

$$\mathcal{L}(x_k, w; \lambda_k) = \mathcal{J}_N(x_k, u_k(w, x_k)) + \frac{1}{2} w^T w + \sum_{k=1}^{N} \lambda_k^T [x_{k+1} - f(x_k, w^T \varphi(x_k))]. \tag{8.27}$$

The conditions for optimality are

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial x_k} = \frac{\partial h}{\partial x_k} + \lambda_{k-1} - (\frac{\partial f}{\partial x_k})^T \lambda_k - (\frac{\partial f}{\partial u_k} \frac{\partial u_k}{\partial x_k})^T \lambda_k = 0, \quad k = 2, ..., N \\ \frac{\partial \mathcal{L}}{\partial x_{N+1}} = \frac{\partial \rho}{\partial x_{N+1}} + \lambda_N = 0 \\ \frac{\partial \mathcal{L}}{\partial w} = w - \sum_{k=1}^{N} \lambda_k^T \frac{\partial f}{\partial u_k} \varphi(x_k) = 0 \\ \frac{\partial \mathcal{L}}{\partial \lambda_k} = x_{k+1} - f(x_k, w^T \varphi(x_k)) = 0, \quad k = 1, ..., N. \end{cases} \tag{8.28}$$

The support values are directly related here to the Lagrange multipliers λ_k. The resulting set of nonlinear equations is of the form

$$F_3(x_k, x_{N+1}, w, \lambda_k) = 0. \tag{8.29}$$

The kernel trick can again be applied, finally yielding a set of equations of the form

$$F_4(x_k, x_{N+1}, \lambda_k) = 0 \tag{8.30}$$

after elimination of w.

8.2.3 *Alternative formulation and stability issues*

An alternative formulation, directly expressed in the unknowns x_k, α_k can be made by considering the optimization problem

$$\min_{x_k, \alpha_k} J_N(x_k, x_k^r, u_k) + \lambda \sum_{k=1}^{N} \alpha_k^2 \tag{8.31}$$

subject to

$$\begin{cases} x_{k+1} &= f(x_k, u_k), \quad x_1 \text{ given} \\ u_k &= \sum_{l=1}^{N} \alpha_l K([x_l; x_l^r], [x_k; x_k^r]) \end{cases}$$

which does not start from the primal weight space but rather specifies the form of the control signal in a dual space. In this formulation a reference state vector x_k^r is considered. A regularization term $\sum_k \alpha_k^2$ is included with λ a positive real constant.

In order to impose local stability at a fixed equilibrium point x^{eq}, one locally linearizes the autonomous closed-loop simulation model by evaluating $\frac{\partial f}{\partial x_k}$ at x^{eq}. LS-SVM control with a local stability constraint can then be formulated as:

$$\min_{x_k, \alpha_k, Q} J_N(x_k, x^{\text{eq}}, u_k) + \lambda \sum_{k=1}^{N} \alpha_k^2 \tag{8.32}$$

subject to

$$\begin{cases} x_{k+1} = f(x_k, \sum_{l=1}^{N} \alpha_l K([x_l; x^{\text{eq}}], [x_k; x^{\text{eq}}])), \ x_1 \text{ given} \\ A^T P A - P < 0 \end{cases}$$

where $P = Q^T Q$, $A = \frac{\partial f}{\partial \hat{x}_k} |_{\hat{x}_k = x^{eq}}$. The last equation is a matrix inequality which expresses that the matrix $A^T P A - P$ is negative definite (maximal eigenvalue negative) meaning that A should be stable [29].

Imposing global asymptotic stability can be done in a somewhat similar way as done in NLq neural control theory [230; 233] or one may impose robust local stability by enlarging the basin of attraction around the equilibrium point according to [212; 284]. Within NLq theory sufficient conditions for global asymptotic stability and input/output stability with finite L_2-gain have been derived, which can also be employed in order to impose robust local stability of the origin for the closed-loop system. Based upon NLq stability criteria, dynamic backpropagation has been modified by imposing matrix inequality constraints. One may proceed in a similar fashion in the case of LS-SVM control laws based upon [212; 284].

When applying an optimization method, the constraints will only hold with a certain tolerance. These small numerical errors may lead to differences between the state vectors as solution to the optimization problem and the simulation of the closed-loop system with LS-SVM control, especially in the control of unstable systems. For control of stable systems this problem will be less critical. A simulation of the closed-loop system with LS-SVM control

$$\hat{x}_{k+1} = f(\hat{x}_k, \sum_{l=1}^{N} \alpha_l K([x_l; x_l^r], [\hat{x}_k; x_k^r])) \qquad \hat{x}_1 = x_1 \text{ given} \tag{8.33}$$

is always needed in order to validate the results, where $\{x_l\}_{l=1}^N$ and $\{\alpha_l\}_{l=1}^N$ are obtained as a solution to the optimization problem.

8.2.4 *Illustrative examples*

A simple state vector tracking example

Here the LS-SVM optimal control method is illustrated on an example reported by Narendra & Mukhopadhyay in [241]. Given the nonlinear system

$$\begin{cases} x_{1,k+1} &= 0.1 x_{1,k} + 2 \frac{u_k + x_{2,k}}{1 + (u_k + x_{2,k})^2} \\ x_{2,k+1} &= 0.1 x_{2,k} + u_k (2 + \frac{u_k^2}{1 + x_{1,k}^2 + x_{2,k}^2}) \end{cases} \tag{8.34}$$

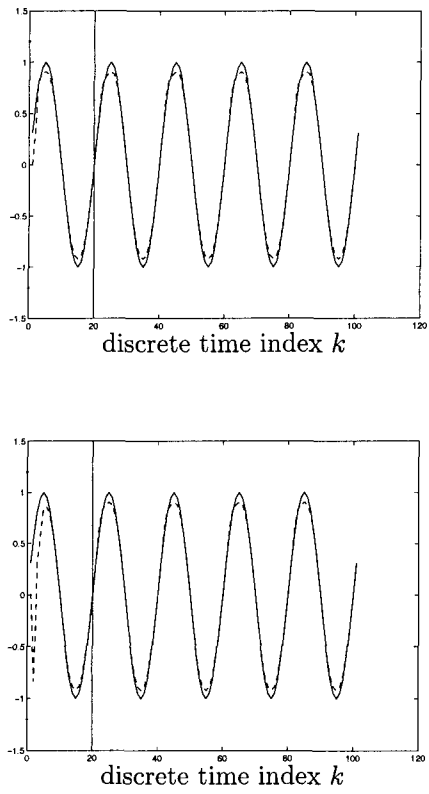

Fig. 8.2 *(Top) Optimal control by LS-SVM with RBF kernel and application of Mercer's theorem. (Full line) first state variable to be tracked; (Dashed line) actual state variable by closed-loop simulation of the system. The data $k = 1, ..., 20$ were used for training of the controller with the origin as initial state. (Bottom) simulation result for another randomly chosen initial state. The LS-SVM controller shows a good generalization performance with respect to other initial states and beyond the time horizon.*

with state variables x_1, x_2 and input u. Consider a state vector tracking problem with

$$\begin{aligned} h(x_k, u_k) &= (x_k - x_k^r)^T Q (x_k - x_k^r) + u_k^T R u_k \\ \rho(x_{N+1}) &= (x_{N+1} - x_{N+1}^r)^T Q (x_{N+1} - x_{N+1}^r) \end{aligned} \tag{8.35}$$

where x_k^r is the reference trajectory to be tracked. Suppose one aims at tracking the first state variable and choose $Q = \mathrm{diag}([1; 0.001])$, $R = 1$, $x_k^r = [\sin(2\pi k/20); \cos(2\pi k/20)]$ with $k = 1, ..., N$ and $N = 20$. The given

initial state vector is $[0;0]$. As control law is taken $w^T\varphi([x_k;x_k^r])$. The derivations are similar to the ones derived for the LS-SVM state feedback control law. In Fig. 8.2 simulation results are shown for the solution obtained from (8.24). The set of nonlinear equations was solved using Matlab's optimization toolbox (function *leastsq*) with unknowns $x_k, u_k, \lambda_k, \alpha_k, \sigma$ (variables e_k eliminated) for $\gamma = 100$ taking an LS-SVM controller with RBF kernel. The unknowns were randomly initialized with zero mean and standard deviation 0.3. The plots show the simulation results for the closed-loop system. The controller is generalizing well towards other initial conditions than the origin (for which it has been trained) and beyond the time horizon of $N = 20$.

Swinging up an inverted pendulum cart-pole system

Control by LS-SVMs is illustrated now on the problem of swinging up an inverted pendulum with local stabilization at the target point. It was shown in [230; 232] that this can also be successfully done by simple MLP architectures.

A nonlinear state space model for the inverted pendulum system (Fig. 8.3) is given by

$$\dot{x} = F(x) + G(x)u \tag{8.36}$$

with

$$F(x) = \begin{bmatrix} x_2 \\ \frac{\frac{4}{3}mlx_4^2 \sin x_3 - \frac{mg}{2}\sin(2x_3)}{\frac{4}{3}m_t - m\cos^2 x_3} \\ x_4 \\ \frac{m_t g \sin x_3 - \frac{ml}{2}x_4^2 \sin(2x_3)}{l(\frac{4}{3}m_t - m\cos^2 x_3)} \end{bmatrix}, \quad G(x) = \begin{bmatrix} 0 \\ \frac{4}{3}\frac{1}{\frac{4}{3}m_t - m\cos^2 x_3} \\ 0 \\ -\frac{\cos x_3}{l(\frac{4}{3}m_t - m\cos^2 x_3)} \end{bmatrix}.$$

The state variables x_1, x_2, x_3, x_4 are the position and velocity of the cart, angle of the pole with the vertical and rate of change of the angle, respectively. The input signal u is the force applied to the cart's center of mass. The symbols m, m_t, l, g denote respectively mass of the pole, total mass of cart and pole, half pole length and the acceleration due to gravity. Here $m = 0.1$, $m_t = 1.1$, $l = 0.5$ is taken. Remark that in the autonomous case $x = [0;0;0;0]$ (pole up) and $x = [0;0;\pi;0]$ (pole down) are equilibrium points. The linearized system around the target equilibrium point (the

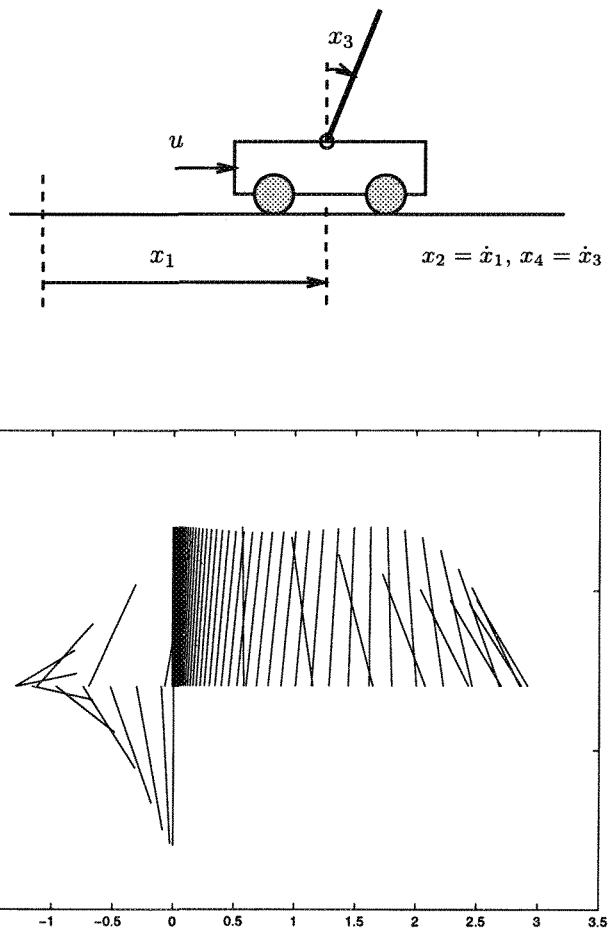

Fig. 8.3 *Swinging up an inverted pendulum by an LS-SVM controller with local stabilization in the upright position. Around the target point the controller is behaving as an LQR controller. (Top) inverted pendulum system; (Bottom) simulation result which visualizes the several pole positions with respect to time.*

origin) is given by

$$\dot{x} = Ax + Bu \tag{8.37}$$

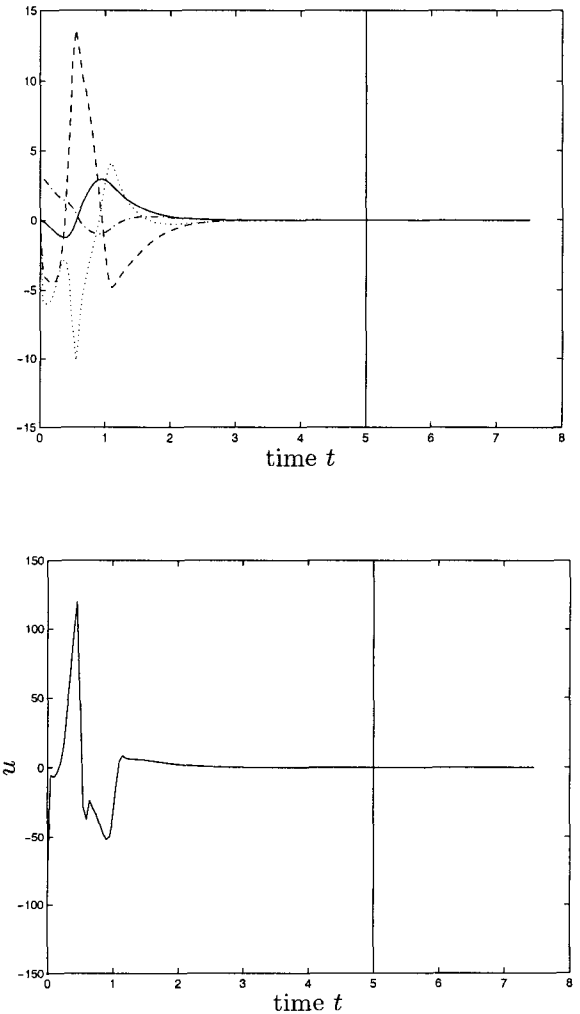

Fig. 8.4 *(Continued) (Top) state variables with respect to time of the closed-loop simulation model with LS-SVM controller: $x_1(t)$ (-); $x_2(t)$ (- -); $x_3(t)$ (-.); $x_4(t)$ (:). The state vector data before the vertical line were used in the training process as support vector data with an RBF kernel; (Bottom) control signal u_k.*

with

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{mg}{\frac{4}{3}m_t - m} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{m_t g}{l(\frac{4}{3}m_t - m)} & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \frac{4}{3}\frac{1}{\frac{4}{3}m_t - m} \\ 0 \\ -\frac{1}{l(\frac{4}{3}m_t - m)} \end{bmatrix}. \tag{8.38}$$

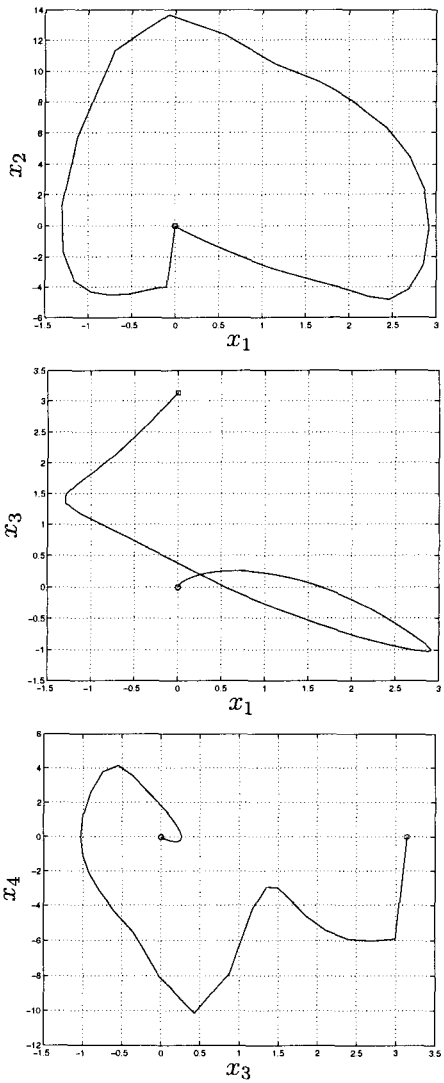

Fig. 8.5 *(Continued) (Top)* $(x_1(t), x_2(t))$; *(Middle)* $(x_1(t), x_3(t))$; *(Bottom)* $(x_3(t), x_4(t))$. *The initial state is marked by a square and the target state by o.*

According to [232] the strategy for the neural controller was to first design a Linear Quadratic Regulator (LQR) controller [28; 85] based upon the linearized model (eventually, also robust linear controllers might be designed,

based on H_∞ control or μ theory, if additional robustness with respect to noise and uncertainties would be needed). This was done in order to impose a set of constraints on the choice of the interconnection weights of a multilayer perceptron controller. The MLP has additional degrees of freedom in comparison with a linear mapping. These additional degrees of freedom are used in order to swing up the pole with local stabilization at the endpoint.

In order to solve the swinging up problem using LS-SVM control, a somewhat similar approach is followed. Based on the continuous time model an LQR controller is designed e.g. for $Q = I$, $R = 0.01$ in the LQR cost function $\int_0^\infty (x^T Q x + u^T R u) dt$ (Matlab command *lqr*). Then the continuous time model has been discretized by using a fourth order Runge-Kutta integration rule with constant step $h_s = 0.05$, resulting in a discrete time model of the form

$$x_{k+1} = \mathcal{F}(x_k, u_k) \tag{8.39}$$

where u_k is assumed to be constant in the time intervals $[kh_s, (k+1)h_s)$ (zero order hold). The control law is chosen as

$$u_k = (L_{\text{lqr}} - L_\Delta) x_k + u_k^{\text{svm}} \tag{8.40}$$

with

$$u_k^{\text{svm}} = \sum_{l=1}^{N} \alpha_l K(x_l, x_k) \tag{8.41}$$

where L_{lqr} is the resulting feedback matrix from LQR design and

$$L_\Delta = \frac{\partial u_k^{\text{svm}}}{\partial x_k} \Big|_{x_k = 0} \tag{8.42}$$

is a modification to the LS-SVM control law such that, locally at the origin, the control law u_k is acting as a LQR controller, in a continuous time sense. The combination between a continuous time LQR result and the discrete time SVM control law may look surprising at first sight, but here it is a convenient way to design an LS-SVM controller for the given continuous time model. An approach according to (8.24) would be more complicated here due to the Runge-Kutta integration rule. An RBF kernel function is taken for the LS-SVM part.

The resulting closed-loop simulation model is given by

$$\hat{x}_{k+1} = \mathcal{F}(\hat{x}_k, (L_{\text{lqr}} - L_\Delta)\hat{x}_k + \sum_{l=1}^{N} \alpha_l K(x_l, \hat{x}_k)) \tag{8.43}$$

with $\hat{x}_1 = x_1 = 0$ given and $\{x_l\}_{l=1}^N$, $\{\alpha_l\}_{l=1}^N$ are the solution to the constrained optimization problem (8.32). As cost function has been taken

$$\mathcal{J}_N = \sum_{k=N-5}^{N} \|x_k\|_2^2 \tag{8.44}$$

with $x_k^r = 0$, $\lambda = 0$ and time horizon $N = 100$ (5 seconds). The constrained optimization problem has been solved using Matlab's optimization toolbox by SQP (Sequential Quadratic Programming) (function *constr*). The values x_k and α_k were initialized by taking small random values. The parameter of the RBF kernel was taken as additional unknown in the optimization problem. In order to emphasize the equations of the constraints, these have been multiplied by a factor 1000. This is needed due to the fact that the system to be controlled is unstable and small differences between x_k (as solution to (8.24)) and $\hat{x}_k$ (in the simulation model (8.33)) may cause large differences otherwise. Simulation results of the closed-loop simulation model are shown in Figs. 8.3-8.5.

Ball and beam control example

Here we discuss the LS-SVM control method on a ball and beam system as described in [241]. The continuous time system description of the ball and beam system (Fig. 8.6) is given by

$$F(x) = \begin{bmatrix} x_2 \\ B(x_1 x_4^2 - G\sin x_3) \\ x_4 \\ 0 \end{bmatrix}, G(x) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{8.45}$$

where $x = [x_1; x_2; x_3; x_4] = [r; \dot{r}; \theta; \dot{\theta}]$ with r the ball position and θ the beam angle and $B = M/(J_b/R_b^2 + M)$ where M, J_b, R_b are the mass, moment of inertia and radius of the ball, respectively. For the control input one has $\tau = 2Mr\dot{r}\dot{\theta} + MGr\cos\theta + (Mr^2 + J + J_b)u$ where τ, G, J denote the torque applied to the beam, the acceleration of gravity and the moment of inertia

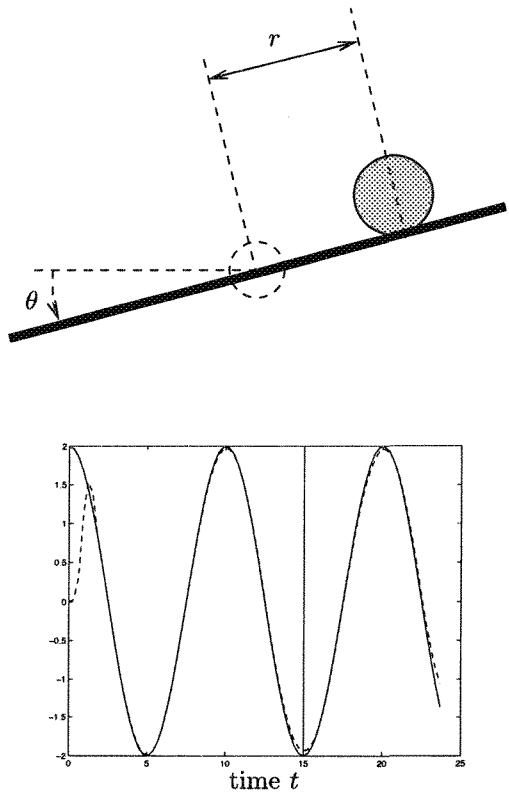

Fig. 8.6 *Tracking control of a ball and beam system by a LS-SVM controller: (Top) ball and beam system; (Bottom) reference input (-) and position of the ball (- -). The state vector data before the vertical line were used in the training process as support vector data with an RBF kernel. Shown is the closed-loop simulation result.*

of the beam, respectively. As control objective we consider tracking of the ball position for a reference input $d(t) = 2\cos(\pi t/5)$.

As for the inverted pendulum example, an LQR design is combined with the LS-SVM control law. The continuous time state space description has been linearized around the origin, for which a LQR controller is designed with $Q = I$, $R = 1$ in the LQR cost function. Then the continuous time model is discretized by using a fourth order Runge-Kutta integration rule with constant step $h_s = 0.3$ resulting in a discrete time model. The first component of the reference state vector $x^r_{1,k}$ is derived from $d(t)$ according

to this step size. The control law is taken as

$$u_k = (L_{\text{lqr}} - L_\Delta) \begin{bmatrix} x_k \\ x^r_{1,k} \end{bmatrix} + u^{\text{svm}}_k \tag{8.46}$$

with

$$u^{\text{svm}}_k = \sum_{l=1}^{N} \alpha_l K(\begin{bmatrix} x_l \\ x^r_{1,l} \end{bmatrix}, \begin{bmatrix} x_k \\ x^r_{1,k} \end{bmatrix}) \tag{8.47}$$

where L_{lqr} is the resulting feedback matrix from LQR (for the autonomous closed-loop system) and

$$L_\Delta = \frac{\partial u^{\text{svm}}_k}{\partial x_k} |_{x_k=0} \, . \tag{8.48}$$

Note that the LS-SVM part also depends on the reference $x^r_{1,k}$. A zero initial state has been taken. SQP was applied for constrained nonlinear optimization with similar initialization as for the inverted pendulum example. Simulation results of the closed-loop simulation model are shown in Fig. 8.6.