Chapter 1
Augmented-SVM for Gradient Observations with Application to Learning Multiple-Attractor Dynamics

Ashwini Shukla and Aude Billard

Abstract In this chapter we present a new formulation that exploits the principle of support vector machine (SVM). This formulation—*Augmented-SVM* (A-SVM)—aims at combining gradient observations with the standard observations of function values (integer labels in classification problems and real values in regression) within a single SVM-like optimization framework. The presented formulation adds onto the existing SVM by enforcing constraints on the gradient of the classifier/regression function. The new constraints modify the original SVM dual, whose optimal solution then results in a new class of support vectors (SV). We present our approach in the light of a particular application in robotics, namely, learning a nonlinear dynamical system (DS) with multiple attractors. Nonlinear DS have been used extensively for encoding robot motions with a single attractor placed at a predefined target where the motion is required to terminate. In this chapter, instead of insisting on a single attractor, we focus on combining several such DS with distinct attractors, resulting in a multi-stable DS. While exploiting multiple attractors provides more flexibility in recovering from unseen perturbations, it also increases the complexity of the underlying learning problem. We address this problem by augmenting the standard SVM formulation with gradient-based constraints derived from the individual DS. The new SV corresponding to the gradient constraints ensure that the resulting multi-stable DS incurs minimum deviation from the original dynamics and is stable at each of the attractors within a finite region of attraction. We show, via implementations on a simulated ten degrees of freedom mobile robotic platform, that the model is capable of real-time motion generation and is able to adapt on-the-fly to perturbations.

A. Shukla (✉) • A. Billard
École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, 1015, Switzerland
e-mail: ashwini.shukla@epfl.ch; aude.billard@epfl.ch

Y. Ma and G. Guo (eds.), *Support Vector Machines Applications*,
DOI 10.1007/978-3-319-02300-7_1,

1.1 Introduction

Dynamical systems (DS) have proved to be a promising framework for encoding and generating complex motions. A major advantage of representing motion using DS-based models [3,10,15,16] is the ability to counter perturbations by virtue of the fact that re-planning of trajectories is instantaneous. These are generative schemes that define the flow of trajectories in state space $\mathbf{x} \in \mathbb{R}^N$ by means of a nonlinear dynamical function $\dot{\mathbf{x}} = f(\mathbf{x})$. DS with single stable attractors have been used in pick and place tasks to control for both the motion of the end-effector [2, 7, 12] and the placement of the fingers on an object [17]. Assuming a single attractor, and hence a single grasping location on the object, constrains considerably the applicability of these methods to realistic grasping problems. A DS composed of multiple stable attractors provides an opportunity to encode different ways to reach and grasp an object. Recent neuro-physiological results [5] have shown that a DS-based modeling best explains the trajectories followed by humans while switching between several reaching targets. From a robotics viewpoint, a robot controlled using a DS with multiple attractors would be able to switch online across grasping strategies. This may be useful, e.g., when one grasping point becomes no longer accessible due to a sudden change in the orientation of the object or the appearance of an obstacle along the current trajectory. Here we present a method using which one can combine—in a single dynamical system—multiple dynamics directed toward different attractors.

The dynamical function $f(\mathbf{x})$ is usually estimated using nonlinear regression functions such as Gaussian Process Regression (GPR) [11], Gaussian Mixture Regression (GMR) [7], and Locally Weighted Projection Regression (LWPR) [13]. However, all of these works modeled DS with a single attractor. While [7,10] ensure global stability at the attractor, other approaches result in unstable DS with spurious attractors.

Stability at multiple targets has been addressed to date largely through neural networks approaches. The Hopfield network and variants offered a powerful means to encode several stable attractors in the same system to provide a form of content-addressable memory [4, 9]. The dynamics to reach these attractors was, however, not controlled for, nor was the partitioning of the state space that would send the trajectories to each attractor. Echo-state networks provide alternative ways to encode various complex dynamics [6]. Although they have proved to be universal estimators, their ability to generalize in untrained regions of state space remains unverified. Also, the key issue of global stability of the learned dynamics is achieved using heuristic rules. To our knowledge, this is the first attempt at learning simultaneously a partitioning of the state space and an embedding of multiple dynamical systems with separate regions of attractions (ROAs) and distinct attractors.

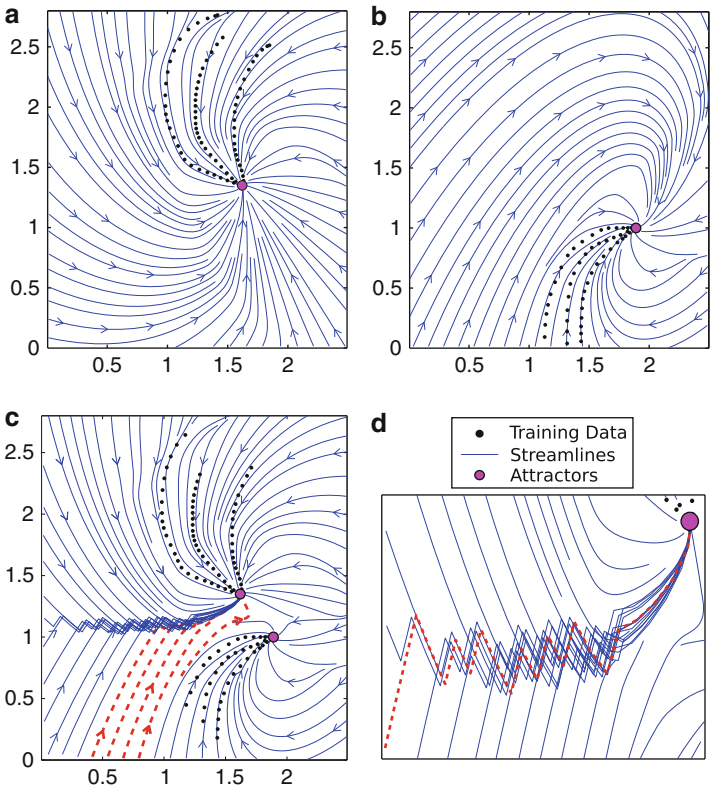

Fig. 1.1 Combining motions using naive SVM classification-based switching. (**a**, **b**)—Two different dynamics with distinct attractors which are to be combined. (**c**)—Employing a simple switching scheme leads to crossing over of some trajectories shown in *red*. (**d**)—Zoomed in around the boundary, showing the fast switching near the boundary

1.2 Identifying Dynamic Constraints

A naive approach to building a multi-attractor DS would be to first partition the space and then learn a DS in each partition separately. This would unfortunately rarely result in the desired compound system. Consider, for instance, two DS with distinct attractors, as shown in Fig. 1.1a, b. First, we build an Support Vector Machine (SVM) classifier to separate data points of the first DS, labeled $+1$, from data points of the other DS, labeled -1. We then estimate each DS separately using any of the techniques reviewed in the previous section. Let $h : \mathbb{R}^N \mapsto \mathbb{R}$ denote the classifier function that separates the state space $\mathbf{x} \in \mathbb{R}^N$ into two regions with labels $y_i \in \{+1, -1\}$. Also, let the two DS be $\dot{\mathbf{x}} = f_{y_i}(\mathbf{x})$ with stable attractors at $\mathbf{x}^*_{y_i}$. The combined DS is then given by $\dot{\mathbf{x}} = f_{\text{sgn}(h(\mathbf{x}))}(\mathbf{x})$. Figure 1.1c shows the trajectories resulting from this approach. Due to the nonlinearity of the dynamics,

trajectories initialized in one region cross the boundary and converge to the attractor located in the opposite region. In other words, each region partitioned by the SVM hyperplane is not a region of attraction for its attractor. In a real-world scenario where the attractors represent grasping points on an object and the trajectories are to be followed by robots, crossing over may take the trajectories towards kinematically unreachable regions. Also, as shown in Fig. 1.1d, trajectories that encounter the boundary may switch rapidly between different dynamics leading to jittery motion.

To ensure that the trajectories do not cross the boundary and remain within the region of attraction of their respective attractors, one could adopt a more informed approach in which each of the original DS is modulated such that the generated trajectories always move away from the classifier boundary. Recall that by construction, the absolute value of the classifier function $h(\mathbf{x})$ increases as one moves away from the classification hyperplane. The gradient $\nabla h(\mathbf{x})$ is hence positive, respectively negative, as one moves inside the region of the positive, respectively negative, class. We can exploit this observation to deflect selective components of the velocity signal from the original DS along, respectively opposite to, the direction $\nabla h(\mathbf{x})$. Concretely, if $\dot{\mathbf{x}}_O = f_{\text{sgn}(h(\mathbf{x}))}(\mathbf{x})$ denotes the velocity obtained from the original DS and

$$\lambda(\mathbf{x}) = \begin{cases} \max\left(\varepsilon, \nabla h(\mathbf{x})^T \dot{\mathbf{x}}_O\right) & \text{if } h(\mathbf{x}) > 0 \\ \min\left(-\varepsilon, \nabla h(\mathbf{x})^T \dot{\mathbf{x}}_O\right) & \text{if } h(\mathbf{x}) < 0 \end{cases}, \tag{1.1}$$

the modulated dynamical system is given by

$$\dot{\mathbf{x}} = \tilde{f}(\mathbf{x}) = \lambda(\mathbf{x}) \nabla h(\mathbf{x}) + \dot{\mathbf{x}}_\perp. \tag{1.2}$$

Here, ε is a small positive scalar and $\dot{\mathbf{x}}_\perp = \dot{\mathbf{x}}_O - \left(\frac{\nabla h(\mathbf{x})^T \dot{\mathbf{x}}_O}{\|\nabla h(\mathbf{x}\|^2}\right) \nabla h(\mathbf{x})$ is the component of the original velocity perpendicular to ∇h. This results in a vector field that flows along increasing values of the classifier function in the regions of space where $h(\mathbf{x}) > 0$ and along decreasing values for $h(\mathbf{x}) < 0$. As a result, the trajectories move away from the classification hyperplane and converge to a point located in the region where they were initialized. Such modulated systems have been used extensively for estimating stability regions of interconnected power networks [8] and are known as *quasi gradient systems* [1]. If $h(\mathbf{x})$ is upper bounded,[1] all trajectories converge to one of the stationary points $\{\mathbf{x} : \nabla h(\mathbf{x}) = 0\}$ and $h(\mathbf{x})$ is a Lyapunov function of the overall system [1, Proposition 1]. Figure 1.2 shows the result of applying the above modulation to our pair of DS. As expected, it forces the trajectories to flow along the gradient of the function $h(\mathbf{x})$. Although this solves the problem of "crossing-over" the boundary, the trajectories obtained are deficient in two major ways. They depart heavily from the original dynamics and do not terminate at the desired attractors. This is due to the fact that the function $h(\mathbf{x})$ used to modulate the DS was designed

[1] SVM classifier function is bounded if the Radial Basis Function (RBF) is used as kernel.

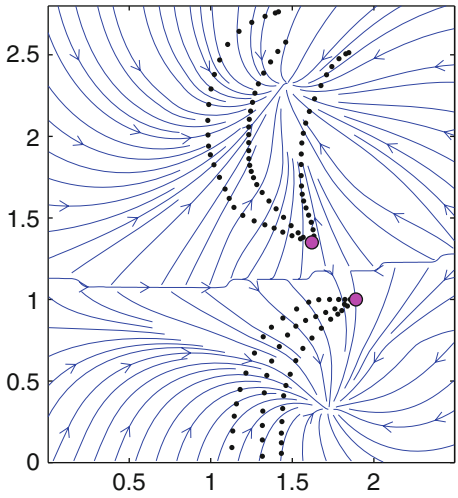

Fig. 1.2 Trajectories obtained by modulating the two original DS with an SVM classifier function. The resulting trajectories flow along directions which register an increase in the value of the classifier function which in turn leads to the bifurcation at the SVM decision boundary

solely for classification and contained no information about the dynamics of the two original DS. In other words, the vector field given by $\nabla h(\mathbf{x})$ was not aligned with the flow of the training trajectories and the stationary points of the modulation function did not coincide with the desired attractors.

In subsequent sections, we show how we can learn a new modulation function which takes into account the three issues we highlighted in this preliminary discussion. We will seek a system that (a) ensures strict classification across ROA for each DS, (b) follows closely the dynamics of each DS in each ROA, and (c) ensures that all trajectories in each ROA reach the desired attractor. Satisfying requirements (a) and (b) above is equivalent to performing classification and regression simultaneously. We take advantage of the fact that the optimization in support vector classification and support vector regression has the same form to phrase our problem in a single constrained optimization framework. In the next sections, we show that in addition to the usual SVM support vectors (SVs), the resulting modulation function is composed of an additional class of SVs. We analyze geometrically the effect of these new support vectors on the resulting dynamics. While this preliminary discussion considered solely binary classification, we will now extend the problem to multi-class classification.

1.3 Problem Formulation

The N-dimensional state space of the system represented by $\mathbf{x} \in \mathbb{R}^N$ is partitioned into M different classes, one for each of the M motions to be combined. We collect trajectories in the state space, yielding a set of P data points $\{\mathbf{x}_i; \dot{\mathbf{x}}_i; l_i\}_{i=1\ldots P}$

where $l_i \in \{1,2,\cdots,M\}$ refers to the class label of each point.[2] To learn the set of modulation functions $\{h_m(\mathbf{x})\}_{m=1\cdots M}$, we proceed recursively. We learn each modulation function in a one-vs-all classifier scheme and then compute the final modulation function $\tilde{h}(\mathbf{x}) = \max_{m=1\cdots M} h_m(\mathbf{x})$. In the multi-class setting, the behavior of avoiding boundaries is obtained if the trajectories move along *increasing* values of the function $\tilde{h}(\mathbf{x})$. To this effect, the deflection term $\lambda(\mathbf{x})$ presented in the binary case Eq. (1.1) becomes $\lambda(\mathbf{x}) = \max\left(\varepsilon, \nabla\tilde{h}(\mathbf{x})^T\dot{\mathbf{x}}_O\right); \forall \mathbf{x} \in \mathbb{R}^N$. Next, we describe the procedure for learning a single $h_m(\mathbf{x})$ function.

We follow the classical SVM formulation and lift the data into a higher dimensional feature space through the mapping $\phi : \mathbb{R}^N \mapsto \mathbb{R}^F$ where F denotes the dimension of the feature space. We also assume that each function $h_m(\mathbf{x})$ is linear in feature space, i.e., $h_m(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b$ where $\mathbf{w} \in \mathbb{R}^F, b \in \mathbb{R}$. We label the current ($m$-th) motion class as positive and all others negative such that the set of labels for the current subproblem is given by

$$y_i = \begin{cases} +1 & \text{if } l_i = m \\ -1 & \text{if } l_i \neq m \end{cases}; \qquad i = 1\cdots P.$$

Also, the set indexing the positive class is then defined as $\mathcal{I}_+ = \{i : i \in [1,P]; l_i = m\}$. With this, we formalize the three constraints explained in Sect. 1.2 as:

Classification — Each point must be classified correctly yields P constraints

$$y_i\left(\mathbf{w}^T\phi(\mathbf{x}_i) + b\right) \geq 1 \quad \forall i = 1\ldots P. \tag{1.3}$$

Lyapunov Constraint — The gradient of the modulation function must have a positive component along the velocities at the data points. This ensures that the modulated flow is aligned with the training trajectories. We have the constraint

$$\nabla h_m(\mathbf{x}_i)^T\hat{\dot{\mathbf{x}}}_i = \mathbf{w}^T\mathbf{J}(\mathbf{x}_i)\hat{\dot{\mathbf{x}}}_i \geq 0 \quad \forall i \in \mathcal{I}_+ \tag{1.4}$$

where $\mathbf{J} \in \mathbb{R}^{F\times N}$ is the Jacobian matrix given by $\mathbf{J} = \left[\nabla\phi_1(\mathbf{x})\nabla\phi_2(\mathbf{x})\cdots\nabla\phi_F(\mathbf{x})\right]^T$ and $\hat{\dot{\mathbf{x}}}_i = \dot{\mathbf{x}}_i/\|\dot{\mathbf{x}}_i\|$ is the normalized velocity at the i-th data point.

Stability — The gradient of the modulation function must vanish at the attractor $\mathbf{x}^*$ of the positive class. This constraint can be expressed as

$$\nabla h_m(\mathbf{x}^*)^T\mathbf{e}_i = \mathbf{w}^T\mathbf{J}(\mathbf{x}^*)\mathbf{e}_i = 0 \quad \forall i = 1\ldots N \tag{1.5}$$

where the set of vectors $\{\mathbf{e}_i\}_{i=1\cdots N}$ is the canonical basis of $\mathbb{R}^N$.

[2] Boldfaced fonts represent vectors. $\mathbf{x}_i$ denotes the i-th vector and x_i denotes the i-th element of vector $\mathbf{x}$.

Incorporating Gradient Observations with Augmented-SVM

Existing variants of the SVM methodology aim at learning **(a)** Classifiers—which satisfy certain *inequality* constraints and **(b)** Regressors—which satisfy *equality* constraints at the data points. Also, in both cases, the constraints are solely defined on the scalar function value. In the A-SVM framework presented in the next sections, we will combine both inequality and equality constraints within the same optimization framework. Moreover, the constraints will be enforced not only on the function value but also on its *gradient*.

1.3.1 Primal and Dual Forms

As in the standard SVM [14], we optimize for maximal margin between the positive and negative classes, subject to constraints (1.3)–(1.5) above. This can be formulated as:

$$\begin{aligned} &\underset{\mathbf{w},\xi_i}{\textbf{minimize}} && \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i\in\mathcal{I}_+}\xi_i \\ &\textbf{subject to} && \left.\begin{aligned} y_i\left(\mathbf{w}^T\phi(\mathbf{x}_i)+b\right) &\geq 1 && \forall i = 1\cdots P \\ \mathbf{w}^T\mathbf{J}(\mathbf{x}_i)\hat{\mathbf{x}}_i + \xi_i &> 0 && \forall i \in \mathcal{I}_+ \\ \xi_i &> 0 && \forall i \in \mathcal{I}_+ \\ \mathbf{w}^T\mathbf{J}(\mathbf{x}^*)\mathbf{e}_i &= 0 && \forall i = 1\cdots N \end{aligned}\right\}. \end{aligned} \tag{1.6}$$

Here $\xi_i \in \mathbb{R}$ are slack variables that relax the Lyapunov constraint in Eq. (1.4). We retain these in our formulation to accommodate noise in the data representing the dynamics. $C \in \mathbb{R}_+$ is a penalty parameter for the slack variables. The Lagrangian for the above problem can be written as

$$\begin{aligned} \mathcal{L}(\mathbf{w},b,\alpha,\beta,\gamma) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i\in\mathcal{I}_+}\xi_i - \sum_{i\in\mathcal{I}_+}\mu_i\xi_i - \sum_{i=1}^{P}\alpha_i\left(y_i(\mathbf{w}^T\phi(\mathbf{x}_i)+b)-1\right) \\ - \sum_{i\in\mathcal{I}_+}\beta_i\left(\mathbf{w}^T\mathbf{J}(\mathbf{x}_i)\hat{\mathbf{x}}_i + \xi_i\right) + \sum_{i=1}^{N}\gamma_i\mathbf{w}^T\mathbf{J}(\mathbf{x}^*)\mathbf{e}_i \end{aligned} \tag{1.7}$$

where $\alpha_i, \beta_i, \mu_i, \gamma_i$ are the Lagrange multipliers with $\alpha_i, \beta_i, \mu_i \in \mathbb{R}_+$, and $\gamma_i \in \mathbb{R}$. Employing a similar analysis as in the standard SVM, we derive the dual by setting the derivatives of the Lagrangian w.r.t all the variables and multipliers to zero, we get

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{P} \alpha_i y_i \phi(\mathbf{x}_i) - \sum_{i \in \mathcal{I}_+} \beta_i \mathbf{J}(\mathbf{x}_i)\hat{\mathbf{x}}_i + \sum_{i=1}^{N} \gamma_i \mathbf{J}(\mathbf{x}^*)\mathbf{e}_i = 0.$$
$$\Rightarrow \mathbf{w} = \sum_{i=1}^{P} \alpha_i y_i \phi(\mathbf{x}_i) + \sum_{i \in \mathcal{I}_+} \beta_i \mathbf{J}(\mathbf{x}_i)\hat{\mathbf{x}}_i - \sum_{i=1}^{N} \gamma_i \mathbf{J}(\mathbf{x}^*)\mathbf{e}_i; \tag{1.8}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{i=1}^{P} \alpha_i y_i = 0; \tag{1.9}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = C - \beta_i - \mu_i = 0 \qquad \forall i \in \mathcal{I}_+. \tag{1.10}$$

Combining (1.10) with the constraints that all the Lagrange multipliers β_i and μ_i be positive, we obtain

$$0 \leq \beta_i \leq C \qquad \forall i \in \mathcal{I}_+. \tag{1.11}$$

Using (1.8), (1.9), and (1.10) in (1.7) we get the dual objective function to be maximized as

$$\hat{\mathcal{L}}(\alpha, \beta, \gamma) = \sum_{i=1}^{P} \alpha_i - \frac{1}{2}\mathbf{w}^T\mathbf{w}. \tag{1.12}$$

Note that although the dual has the same general form as the dual in the standard SVM formulation, it differs in the expression of the term $\mathbf{w}$. Expanding using (1.8) we have

$$\begin{aligned}
\mathbf{w}^T\mathbf{w} = & \left(\sum_{i=1}^{P} \alpha_i y_i \phi(\mathbf{x}_i)^T + \sum_{i \in \mathcal{I}_+} \beta_i \hat{\mathbf{x}}_i^T \mathbf{J}(\mathbf{x}_i)^T - \sum_{i=1}^{N} \gamma_i \mathbf{e}_i^T \mathbf{J}(\mathbf{x}^*)^T \right) \\
& \left(\sum_{j=1}^{P} \alpha_j y_j \phi(\mathbf{x}_j) + \sum_{j \in \mathcal{I}_+} \beta_j \mathbf{J}(\mathbf{x}_j)\hat{\mathbf{x}}_j - \sum_{j=1}^{N} \gamma_j \mathbf{J}(\mathbf{x}^*)\mathbf{e}_j \right) \\
= & \sum_{i=1}^{P} \left(\sum_{j=1}^{P} \alpha_i y_i \alpha_j y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) + \sum_{j \in \mathcal{I}_+} \alpha_i y_i \beta_j \phi(\mathbf{x}_i)^T \mathbf{J}(\mathbf{x}_j)\hat{\mathbf{x}}_j - \sum_{j=1}^{N} \alpha_i y_i \gamma_j \phi(\mathbf{x}_i)^T \mathbf{J}(\mathbf{x}^*)\mathbf{e}_j \right) \\
& + \sum_{i \in \mathcal{I}_+} \left(\sum_{j=1}^{P} \beta_i \alpha_j y_j \hat{\mathbf{x}}_i^T \mathbf{J}(\mathbf{x}_i)^T \phi(\mathbf{x}_j) + \sum_{j \in \mathcal{I}_+} \beta_i \beta_j \hat{\mathbf{x}}_i^T \mathbf{J}(\mathbf{x}_i)^T \mathbf{J}(\mathbf{x}_j)\hat{\mathbf{x}}_j - \sum_{j=1}^{N} \beta_i \gamma_j \hat{\mathbf{x}}_i^T \mathbf{J}(\mathbf{x}_i)^T \mathbf{J}(\mathbf{x}^*)\mathbf{e}_j \right) \\
& - \sum_{i=1}^{N} \left(\sum_{j=1}^{P} \gamma_i \alpha_j y_j \mathbf{e}_i^T \mathbf{J}(\mathbf{x}^*)^T \phi(\mathbf{x}_j) + \sum_{j \in \mathcal{I}_+} \gamma_i \beta_j \mathbf{e}_i^T \mathbf{J}(\mathbf{x}^*)^T \mathbf{J}(\mathbf{x}_j)\hat{\mathbf{x}}_j - \sum_{j=1}^{N} \gamma_i \gamma_j \mathbf{e}_i^T \mathbf{J}(\mathbf{x}^*)^T \mathbf{J}(\mathbf{x}^*)\mathbf{e}_j \right).
\end{aligned} \tag{1.13}$$

Rewriting in matrix form,

$$\mathbf{w}^T\mathbf{w} = \begin{bmatrix}\alpha^T & \beta^T & \gamma^T\end{bmatrix}\begin{bmatrix}\mathrm{K} & \mathrm{G} & -\mathrm{G}_* \\ \mathrm{G}^T & \mathrm{H} & -\mathrm{H}_* \\ -\mathrm{G}_*^T & -\mathrm{H}_*^T & \mathrm{H}_{**}\end{bmatrix}\begin{bmatrix}\alpha \\ \beta \\ \gamma\end{bmatrix} \tag{1.14}$$

where $\mathrm{K} \in \mathbb{R}^{P\times P}, \mathrm{G} \in \mathbb{R}^{P\times|\mathcal{I}_+|}, \mathrm{G}_* \in \mathbb{R}^{P\times N}, \mathrm{H} \in \mathbb{R}^{|\mathcal{I}_+|\times|\mathcal{I}_+|}, \mathrm{H}_* \in \mathbb{R}^{|\mathcal{I}_+|\times N}, \mathrm{H}_{**} \in \mathbb{R}^{N\times N}$ are given by

$$\left.\begin{aligned}[\mathrm{K}]_{ij} &= y_i y_j \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) \;; & [\mathrm{H}]_{ij} &= \hat{\mathbf{x}}_i^T \mathrm{J}(\mathbf{x}_i)^T \mathrm{J}(\mathbf{x}_j)\hat{\mathbf{x}}_j \\ [\mathrm{G}]_{ij} &= y_i \phi(\mathbf{x}_i)^T \mathrm{J}(\mathbf{x}_j)\hat{\mathbf{x}}_j \;; & [\mathrm{H}_*]_{ij} &= \hat{\mathbf{x}}_i^T \mathrm{J}(\mathbf{x}_i)^T \mathrm{J}(\mathbf{x}^*)\mathbf{e}_j \\ [\mathrm{G}_*]_{ij} &= y_i \phi(\mathbf{x}_i)^T \mathrm{J}(\mathbf{x}^*)\mathbf{e}_j \;; & [\mathrm{H}_{**}]_{ij} &= \mathbf{e}_i^T \mathrm{J}(\mathbf{x}^*)^T \mathrm{J}(\mathbf{x}^*)\mathbf{e}_j\end{aligned}\right\}. \tag{1.15}$$

where $[.]_{ij}$ denotes the i,j-th entry of the corresponding matrix. Further using the relations (1.19) and (1.20) from Appendix 1, we can rewrite the above block matrices in terms of the kernel function and data:

$$\left.\begin{aligned}[\mathrm{K}]_{ij} &= y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) \quad\;\; ; & [\mathrm{H}]_{ij} &= \hat{\mathbf{x}}_i^T \frac{\partial^2 k(\mathbf{x}_i,\mathbf{x}_j)}{\partial \mathbf{x}_i \partial \mathbf{x}_j}\hat{\mathbf{x}}_j \\ [\mathrm{G}]_{ij} &= y_i \left(\frac{\partial k(\mathbf{x}_i,\mathbf{x}_j)}{\partial \mathbf{x}_j}\right)^T \hat{\mathbf{x}}_j \;; & [\mathrm{H}_*]_{ij} &= \hat{\mathbf{x}}_i^T \frac{\partial^2 k(\mathbf{x}_i,\mathbf{x}^*)}{\partial \mathbf{x}_i \partial \mathbf{x}^*}\mathbf{e}_j \\ [\mathrm{G}_*]_{ij} &= y_i \left(\frac{\partial k(\mathbf{x}_i,\mathbf{x}^*)}{\partial \mathbf{x}^*}\right)^T \mathbf{e}_j \;; & [\mathrm{H}_{**}]_{ij} &= \mathbf{e}_i^T \frac{\partial^2 k(\mathbf{x}^*,\mathbf{x}^*)}{\partial \mathbf{x}^* \partial \mathbf{x}^*}\mathbf{e}_j\end{aligned}\right\}. \tag{1.16}$$

These can be further expanded given a choice of the kernel. Expansions for the RBF and the nonhomogeneous polynomial kernel are given in Appendix 2.

Using Eqs. (1.9), (1.11), (1.12), and (1.14) the dual optimization problem can be stated as

$$\underset{\alpha,\beta,\gamma}{\textbf{minimize}}\ \frac{1}{2}\begin{bmatrix}\alpha^T\beta^T\gamma^T\end{bmatrix}\begin{bmatrix}\mathrm{K} & \mathrm{G} & -\mathrm{G}_* \\ \mathrm{G}^T & \mathrm{H} & -\mathrm{H}_* \\ -\mathrm{G}_*^T & -\mathrm{H}_*^T & \mathrm{H}_{**}\end{bmatrix}\begin{bmatrix}\alpha \\ \beta \\ \gamma\end{bmatrix} - \alpha^T\bar{\mathbf{1}}$$

subject to

$$\left.\begin{aligned} 0 \le \alpha_i \quad\quad & \forall i = 1\cdots P \\ 0 \le \beta_i \le C \quad\quad & \forall i \in \mathcal{I}_+ \\ \textstyle\sum_{i=1}^P \alpha_i y_i = 0 \quad\quad & \end{aligned}\right\}. \tag{1.17}$$

Note that the Lagrange multipliers γ_i are completely unconstrained as they correspond to the *equality* constraints in the primal. Also, since the matrices K, H, and H_{**} are symmetric, the overall Hessian matrix for the resulting quadratic program is also symmetric. In our implementation, we use the MATLAB® *quadprog* solver to solve this quadratic program. We initialize the iterations by setting α_i as the solution to the standard SVM classification problem. All β_i and γ_i are set to zeros. Once the optimal solution for the above problem is obtained, the modulation function can be written as

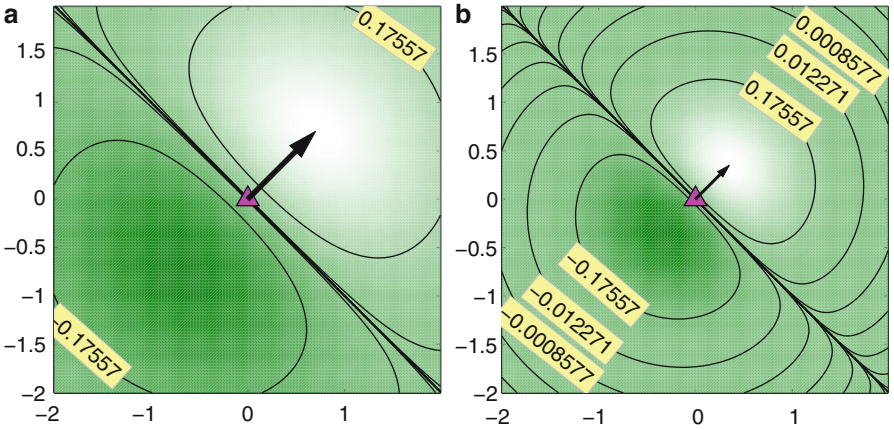

Fig. 1.3 Isocurves of $f(\mathbf{x}) = \hat{\mathbf{x}}_i^T \frac{\partial k(\mathbf{x},\mathbf{x}_i)}{\partial \mathbf{x}_i}$ at $\mathbf{x}_i = [0\,0]^T, \hat{\mathbf{x}}_i = [\frac{1}{\sqrt{2}}\,\frac{1}{\sqrt{2}}]^T$ for the RBF kernel with width σ. (**a**) $\sigma = 1$. (**b**) $\sigma = 0.5$

$$
\begin{aligned}
h(\mathbf{x}) &= \mathbf{w}^T \phi(\mathbf{x}) + b \\
&= \sum_{i=1}^{P} \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + \sum_{i \in \mathcal{I}_+} \beta_i \hat{\mathbf{x}}_i^T \mathbf{J}(\mathbf{x}_i)^T \phi(\mathbf{x}) - \sum_{i=1}^{N} \gamma_i \mathbf{e}_i^T \mathbf{J}(\mathbf{x}^*)^T \phi(\mathbf{x}) + b \\
&= \sum_{i=1}^{P} \alpha_i y_i k(\mathbf{x},\mathbf{x}_i) + \sum_{i \in \mathcal{I}_+} \beta_i \dot{\mathbf{x}}_i^T \frac{\partial k(\mathbf{x},\mathbf{x}_i)}{\partial \mathbf{x}_i} - \sum_{i=1}^{N} \gamma_i \mathbf{e}_i^T \frac{\partial k(\mathbf{x},\mathbf{x}^*)}{\partial \mathbf{x}^*} + b
\end{aligned}
\tag{1.18}
$$

This modulation function has noticeable similarities with the standard SVM classifier function. The first summation term on the right-hand side is composed of the α support vectors (α-SV) which act as support to the classification hyperplane. The second term entails a new class of support vectors that perform a linear combination of the normalized velocity $\hat{\mathbf{x}}_i$ at the training data points $\mathbf{x}_i$. These β support vectors (β-SVs) collectively contribute to the fulfilment of the Lyapunov constraint in Eq. (1.4) by introducing a positive slope in the modulation function value along the directions $\hat{\mathbf{x}}_i$. Figure 1.3 shows the influence of a single β-SV for the RBF kernel $k(\mathbf{x}_i,\mathbf{x}_j) = e^{-1/2\sigma^2 \|\mathbf{x}_i - \mathbf{x}_j\|^2}$ with $\mathbf{x}_i$ at the origin and $\hat{\mathbf{x}}_i = [\frac{1}{\sqrt{2}}\,\frac{1}{\sqrt{2}}]^T$. Observe that the smaller the kernel width σ, the steeper the slope. The third summation term is a nonlinear bias, which does not depend on the chosen support vectors, and performs a local modification around the desired attractor $\mathbf{x}^*$ to ensure that the modulation function has a local maximum at that point. b is the constant bias which normalizes the classification margins as -1 and $+1$. We calculate its value by making use of the fact that for all the data points $\mathbf{x}_i$ chosen as α-SV, we must have $y_i h_m(\mathbf{x}_i) = 1$. We use the average of the values obtained from different support vectors.

Figure 1.4 illustrates the effects of the support vectors in a 2D example by progressively adding them and overlaying the resulting DS flow in each case.

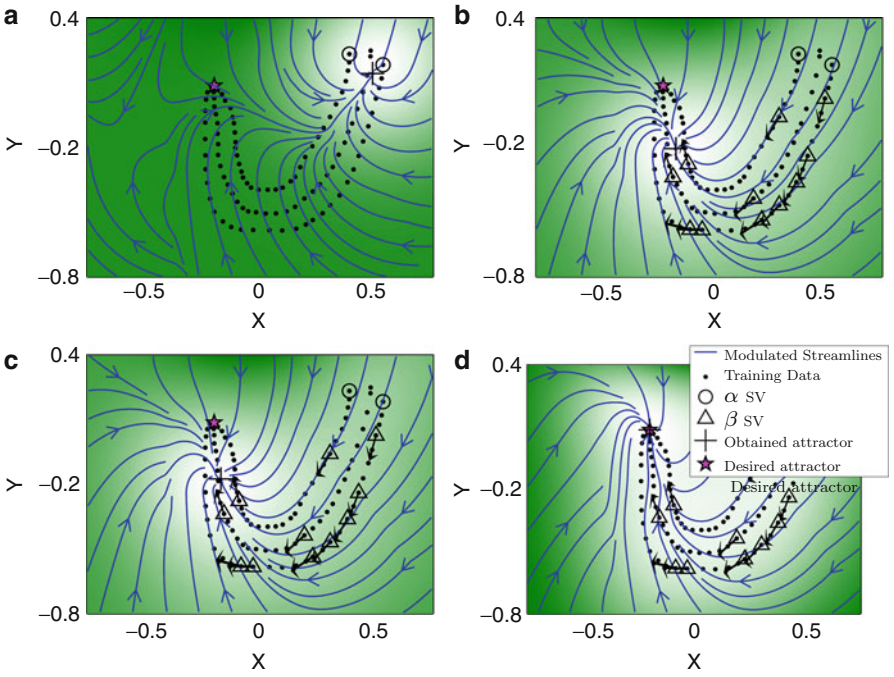

Fig. 1.4 Progressively adding support vectors to highlight their effect on shaping the dynamics of the motion. (**a**) α—SVs largely affect classification. (**b**)—β-SVs guide the flow of trajectories along their respective associated directions $\hat{\mathbf{x}}_i$ shown by *arrows*. (**c**, **d**) The two γ terms force the local maximum of the modulation function to coincide with the desired attractor along the X and Y axes respectively

The value of the modulation function $h_m(\mathbf{x})$ is shown by the color plot (white indicates high values). As the β-SVs are added—as shown in Fig. 1.4b—they *force* the flow of trajectories along their associated directions. In Fig. 1.4c, d, adding the two γ terms shifts the location of the maximum of the modulation function to coincide with the desired attractor. Once all the SVs have been taken into account, the streamlines of the resulting DS achieve the desired criteria, i.e., they follow the training trajectories and terminate at the desired attractor.

1.4 Application Examples

In this section, we validate the presented A-SVM model on 2D (synthetic) data and on a robotic simulated experiment using a seven degrees of freedom (DOF) KUKA-LWR arm mounted on a three-DOF Omnirob base to catch falling objects. A video of the robotic experiment—simulated and real—is provided at the project url http://asvm.epfl.ch. Next, we present a cross-validation analysis of the error

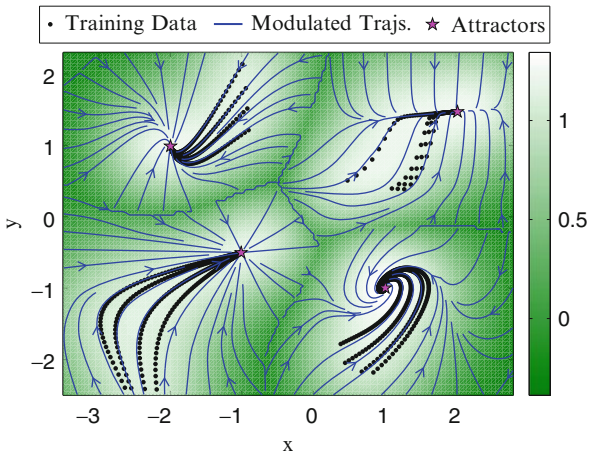

Fig. 1.5 Resulting flow of the synthetic four-attractor example. Color plot depicts the value of the one-vs-all classifier function which has local maximums at all the four attractors

introduced by the modulation in the original dynamics. A sensitivity analysis of the region of attraction of the resulting dynamical system with respect to the model parameters is also presented. We used the RBF kernel for all the results presented in this section. As discussed in Sect. 1.2, the RBF kernel is advantageous as it ensures that the function $h_m(\mathbf{x})$ is bounded. To generate an initial estimate of each individual dynamical system, we used the technique proposed in Khansari-Zadeh and Billard [7].

1.4.1 2D Example

Figure 1.5 shows a synthetic example with four motion classes, each generated from a different closed form dynamics and containing 160 data points. The color plot indicates the value of the combined modulation function $\tilde{h}(\mathbf{x}) = \max_{m=1\cdots M} h_m(\mathbf{x})$ where each of the functions $h_m(\mathbf{x})$ is learned using the presented A-SVM technique. A total of nine support vectors were obtained which is $<10\,\%$ of the number of training data points. The trajectories obtained after modulating the original dynamical systems flow along increasing values of the modulation function, thereby bifurcating towards different attractors at the region boundaries. Unlike the dynamical system in Fig. 1.2, the flow here is aligned with the training trajectories and terminates at the desired attractors. To recall, this is made possible thanks to the additional constraints Eqs. (1.4) and (1.5) in our formulation.

In a second example, we tested the ability of our model to accommodate a higher density of attractors. We created eight synthetic dynamics by capturing motion data using a screen mouse. Figure 1.6 shows the resulting eight-attractor system.

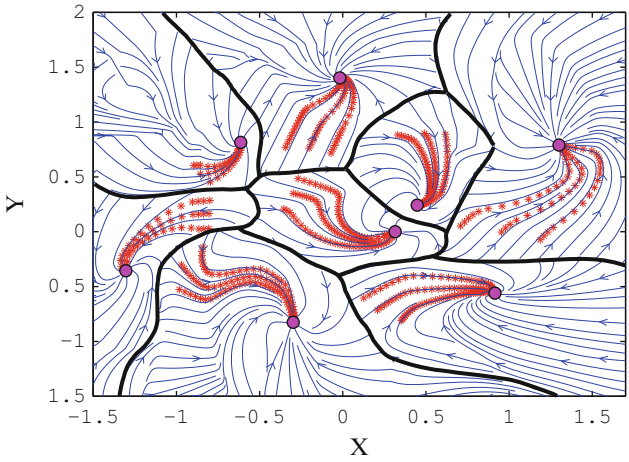

Fig. 1.6 A DS with eight attractors learned using only a few representative trajectories (*black dots*) from each DS. The bifurcation boundaries, as well as the dynamics of each DS need to be estimated from these trajectories

1.4.2 Error Analysis

As formulated in Eq. (1.6), the Lyapunov constraints admit some slack, which allows the modulation to introduce slight deviations from the original dynamics. Here we statistically analyze this error via fivefold cross validation. In the four-attractor problem presented above, we generate a total of ten trajectories per motion class and use 2:3 training to testing ratio for cross validation.

We calculate the average percentage error between the original velocity (read off from the data) and the modulated velocity [calculated using Eq. (1.2)] for the m-th class as $e_m = \left\langle \frac{\|\dot{\mathbf{x}}_i - \tilde{f}(\mathbf{x}_i)\|}{\|\dot{\mathbf{x}}_i\|} \times 100 \right\rangle_{i:l_i=m}$ where $< . >$ denotes average over the indicated range. Figure 1.7a shows the cross-validation error (mean and standard deviation over the fivefolds) for a range of values of kernel width. The general trend revealed here is that for each class of motion, there exists a band of optimum values of the kernel width for which the testing error is the smallest. The region covered by this band of optimal values may vary depending on the relative location of the attractors and other data points. In Fig. 1.5, motion classes 2 (upper left) and 4 (upper right) are better fitted and show less sensitivity to the choice of kernel width than classes 1 (lower left) and 3 (lower right). We will show later in this section that this is correlated with the distance between the attractors. A comparison of testing and training errors for the least error case is shown in Fig. 1.7b. We see that the testing errors for all the classes in the best case scenario are less than 1 %.

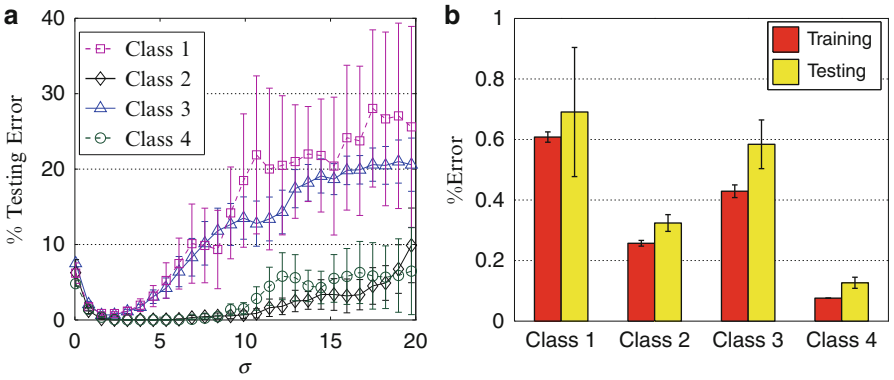

Fig. 1.7 Error analysis for the synthetic four-attractor example. (**a**) Cross-validation error. (**b**) Best case errors

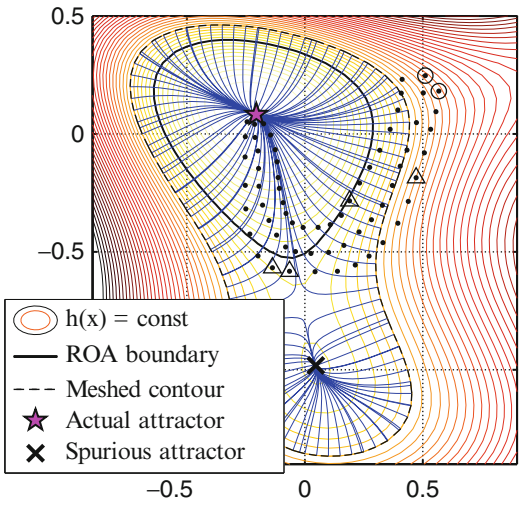

Fig. 1.8 Test trajectories starting from several points on an isocurve (*dotted line*) to determine spurious attractors

1.4.3 Sensitivity Analysis

The partitioning of space created by our method results in M ROA for each of our M attractors. To assess the size of these regions and the existence of spurious attractors, we adopt an empirical approach. For each class, we compute the isosurfaces of the corresponding modulation function $h_m(\mathbf{x})$ in the range $[0, h_m(\mathbf{x}^*)]$. These hypersurfaces incrementally span the volume of the m-th region around its attractor. We mesh each of these test surfaces and compute trajectories starting from the obtained mesh-points, looking for spurious attractors. h_{ROA} is the isosurface of maximal value that encloses no spurious attractor and marks the ROA of the corresponding motion dynamics. We use the example in Fig. 1.4 to illustrate this process. Figure 1.8 shows a case where one spurious attractor is detected using a

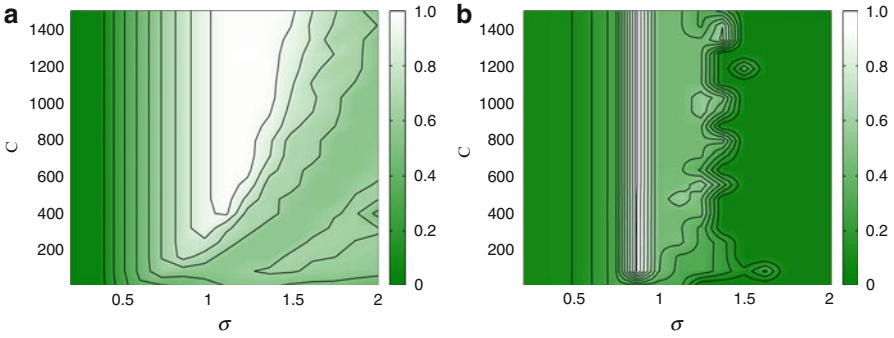

Fig. 1.9 Variation of r_{ROA} with varying model parameters. (**a**) $d_{att} = 1.0$. (**b**) $d_{att} = 0.2$

larger test surface (*dotted line*) whereas the actual ROA (*solid line*) is smaller. Once h_{ROA} is calculated, we define the size of ROA as $r_{ROA} = (h(\mathbf{x}^*) - h_{ROA})/h(\mathbf{x}^*)$. $r_{ROA} = 0$ when no trajectory except those originating at the attractor itself lead to the attractor. $r_{ROA} = 1$ when the ROA is bounded by the isosurface $h(\mathbf{x}) = 0$. The size of the r_{ROA} is affected by both the choice of kernel width and the distance across nearby attractors. This is illustrated in Fig. 1.9 using data points from class 1 of Fig. 1.5 and translating the attractors so that they are either very far apart (left, distance $d_{att} = 1.0$) or very close to one another (right, $d_{att} = 0.2$). As expected, r_{ROA} increases as we reach the optimal range of parameters. Furthermore, when the attractors are farther apart, high values of r_{ROA} are obtained for a larger range of values of the kernel width, i.e., the model is less sensitive to the chosen kernel width. With smaller distance between the attractors (Fig. 1.9b), only a small deviation from the optimum kernel width results in a considerable loss in r_{ROA}, exhibiting high sensitivity to the model parameter.

1.4.4 3D Example

We validated our method on a real-world 3D problem. The attractors here represent manually labeled grasping points on a pitcher. The 3D model of the object was taken from the ROS IKEA object library. We use the seven-DOF KUKA-LWR arm mounted on the three-DOF KUKA-Omnirob base for executing the modulated Cartesian trajectories in simulation. We control all ten DOF of the robot using the damped least square inverse kinematics. Training data for this implementation was obtained by recording the end-effector positions $\mathbf{x}_i \in \mathbb{R}^3$ from kinesthetic demonstrations of reach-to-grasp motions directed towards these grasping points, yielding a three-class problem (see Fig. 1.10a). Each class was represented by 75 data points. Figure 1.10b shows the isosurfaces $h_m(\mathbf{x}) = 0; m \in \{1,2,3\}$ learned using the presented method. Figure 1.11a, b show the robot executing two trajectories when started from two different locations and converging to a different attractor (grasping point). Figure 1.10c shows the flow of motion around the object. Note

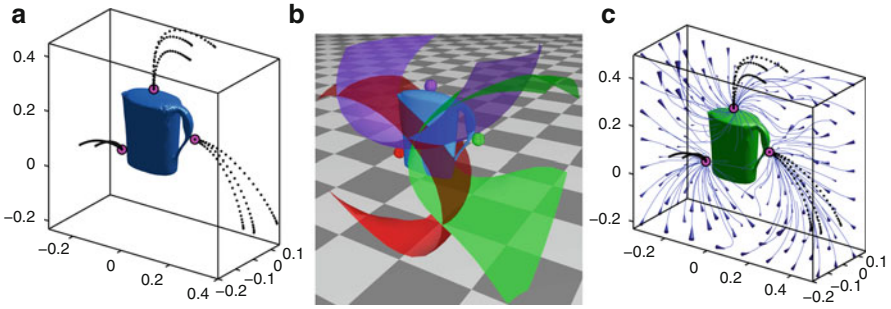

Fig. 1.10 3D Experiment. (**a**) shows training trajectories for three manually chosen grasping points. (**b**) shows the isosurfaces $h_m(\mathbf{x}) = 0; m = 1, 2, 3$ along with the locations of the corresponding attractors. (**c**) shows the complete flow of motion

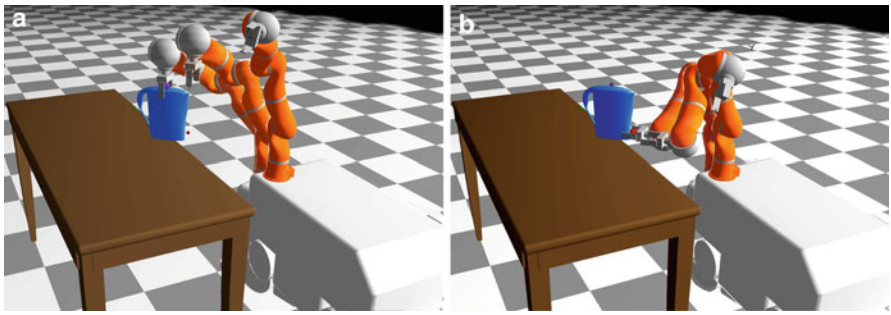

Fig. 1.11 Ten-DOF mobile robot executes the generated trajectories starting from different positions and hence converging to different grasping points (attractors). (**a**) Trajectory 1. (**b**) Trajectory 2

that the time required to generate each trajectory point is $O(S)$ where S denotes the total number of support vectors in the model. In this particular example with a total of 18 SVs, the trajectory points were generated at 1,000 Hz which is well suited for real-time control. Such a fast generative model allows the robot to switch on-the-fly between the attractors and adapt to real-time perturbations in the object or the end-effector pose, without any re-planning or re-learning. Results for another object—a champagne glass with two attractors—are shown in Fig. 1.12. We performed high speed experiments in which the glass is falling and the robot needs to catch it at one of the two attractors. This requires real-time adaptation to the constantly changing position and orientation of the object. The robot might need to switch between the attractors and move the end-effector toward the chosen attractor. Figure 1.13 shows the experiments in simulation and with the real KUKA robot. Full videos explaining the A-SVM methodology and these experiments are available at http://asvm.epfl.ch/download.php.

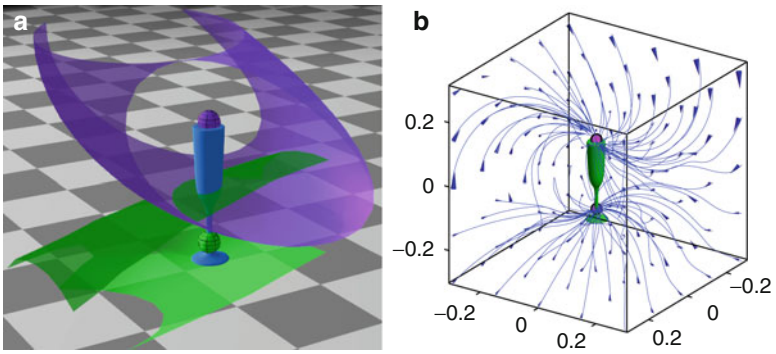

Fig. 1.12 (**a**) Two attractors placed on a champagne glass and their corresponding classification surfaces. (**b**) Complete flow of motion around the object

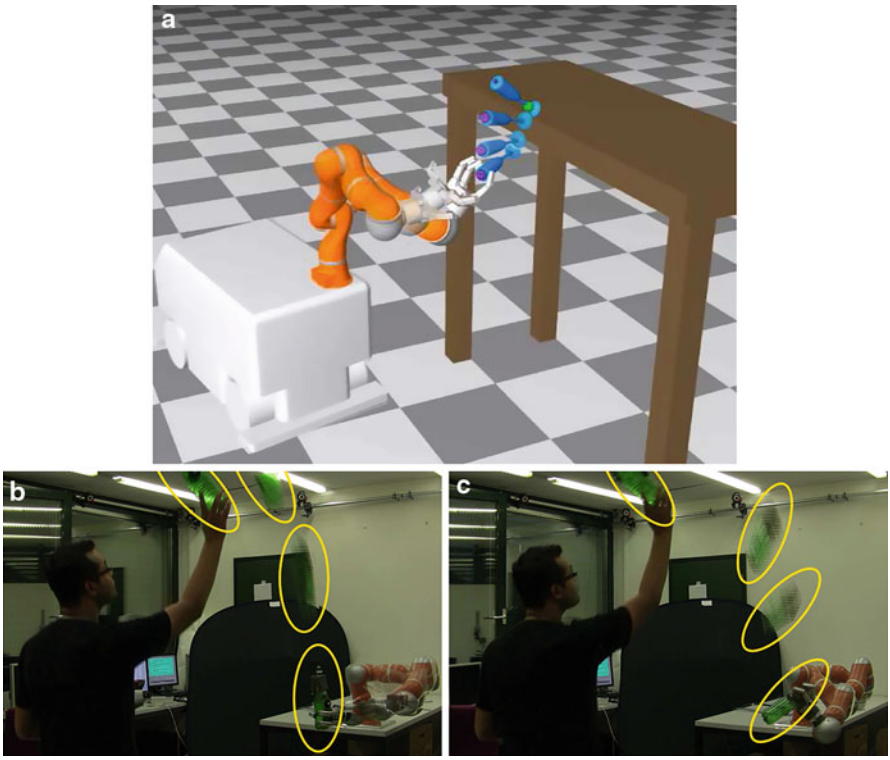

Fig. 1.13 (**a**)—Simulation experiment of catching a falling object with the ten-DOF KUKA-Omnirob. The robot switches between attractors (*green* to *magenta*) as the object falls down. (**b**, **c**)—Real seven-DOF KUKA arm catching the falling object at different grasping points (attractors) in different throwing situations

1.5 Conclusions

We presented the A-SVM model for combining nonlinear dynamical systems through a partitioning of the space. We reformulated the optimization framework of SVM to encapsulate gradient-based constraints that ensure accurate reproduction of the dynamics of motion. The new set of constraints result in a new class of support vectors that exploit partial derivatives of the kernel function to align the flow of trajectories with the training data. The resulting model behaves as a multi-stable DS with attractors at the desired locations. Each of the classified regions is forward invariant w.r.t the learned DS. This ensures that the trajectories do not cross over region boundaries. We validated the presented method on synthetic motions in 2D and 3D grasping motions on real objects. Results show that even though spurious attractors may occur, in practice they can be avoided by a careful choice of model parameters through grid search. The applicability of the method for real-time control of a ten-DOF robot was also demonstrated.

Acknowledgments This work was supported by EU Project *First-MM* (FP7/2007–2013) under grant agreement number *248258*. The authors would also like to thank Prof. François Margot for his insightful comments on the technical material.

Appendix 1: Kernel Derivatives

For scalar variables $x_i, x_j \in \mathbb{R}$ and any feature transformation $\phi : \mathbb{R} \mapsto \mathbb{R}^F$ we define a valid Mercer kernel as $k(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$. If $\prime$ denotes the derivative w.r.t the state variable, then the identities $\phi'(x_i)^T \phi(x_j) = \frac{\partial k(x_i,x_j)}{\partial x_i}$ and $\phi'(x_i)^T \phi'(x_j) = \frac{\partial^2 k(x_i,x_j)}{\partial x_i \partial x_j}$ follow directly from the definition of the kernel. We can rewrite these identities for vector variables $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^{\mathbb{N}}$ by taking the derivative w.r.t one of the components (say n-th) as $\left(\frac{\partial \phi(\mathbf{x}_i)}{\partial \mathbf{x}(n)}\right)^T \phi(\mathbf{x}_j) = \frac{\partial k(\mathbf{x}_i,\mathbf{x}_j)}{\partial \mathbf{x}_i(n)}$. Expanding the first vector term we get

$$\Rightarrow \left[\frac{\partial \phi_1(\mathbf{x}_i)}{\partial \mathbf{x}(n)}, \frac{\partial \phi_2(\mathbf{x}_i)}{\partial \mathbf{x}(n)}, \cdots, \frac{\partial \phi_F(\mathbf{x}_i)}{\partial \mathbf{x}(n)}\right] \phi(\mathbf{x}_j) = \frac{\partial k(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i(n)}.$$

Stacking the above equation in rows for $n = 1 \ldots N$, we get

$$\begin{bmatrix} \frac{\partial \phi_1(\mathbf{x}_i)}{\partial \mathbf{x}(1)} & \frac{\partial \phi_2(\mathbf{x}_i)}{\partial \mathbf{x}(1)} & \cdots & \frac{\partial \phi_F(\mathbf{x}_i)}{\partial \mathbf{x}(1)} \\ \frac{\partial \phi_1(\mathbf{x}_i)}{\partial \mathbf{x}(2)} & \frac{\partial \phi_2(\mathbf{x}_i)}{\partial \mathbf{x}(2)} & \cdots & \frac{\partial \phi_F(\mathbf{x}_i)}{\partial \mathbf{x}(2)} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \phi_1(\mathbf{x}_i)}{\partial \mathbf{x}(N)} & \frac{\partial \phi_2(\mathbf{x}_i)}{\partial \mathbf{x}(N)} & \cdots & \frac{\partial \phi_F(\mathbf{x}_i)}{\partial \mathbf{x}(N)} \end{bmatrix} \phi(\mathbf{x}_j) = \begin{bmatrix} \frac{\partial k(\mathbf{x}_i,\mathbf{x}_j)}{\partial \mathbf{x}_i(1)} \\ \frac{\partial k(\mathbf{x}_i,\mathbf{x}_j)}{\partial \mathbf{x}_i(2)} \\ \vdots \\ \frac{\partial k(\mathbf{x}_i,\mathbf{x}_j)}{\partial \mathbf{x}_i(N)} \end{bmatrix}$$

$$\Rightarrow \mathrm{J}(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = \frac{\partial k(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i} \tag{1.19}$$

where J denotes the standard Jacobian matrix for a vector valued function. Similarly, by writing the derivatives w.r.t (n,m)-th dimension and putting them as the corresponding element of a Hessian matrix we get

$$\mathrm{J}(\mathbf{x}_i)^T \mathrm{J}(\mathbf{x}_j) = \frac{\partial^2 k(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i \partial \mathbf{x}_j}. \tag{1.20}$$

Appendix 2: Specific Kernel Expansions

The above formulation is generic and can be applied to any kernel. Here we give the RBF kernel-specific expressions for the block matrices in (1.15).

RBF Kernel

$$[\mathrm{K}]_{ij} = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) = y_i y_j e^{-d\|\mathbf{x}_i - \mathbf{x}_j\|^2}$$

$$[\mathrm{G}]_{ij} = y_i \left(\frac{\partial k(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_j} \right)^T \hat{\mathbf{x}}_j = -2d y_i e^{-d\|\mathbf{x}_i - \mathbf{x}_j\|^2} (\mathbf{x}_j - \mathbf{x}_i)^T \hat{\mathbf{x}}_j$$

Replacing $\mathbf{x}_j$ by $\mathbf{x}^*$ in the above equation we get

$$[\mathrm{G}_*]_{ij} = y_i \left(\frac{\partial k(\mathbf{x}_i, \mathbf{x}^*)}{\partial \mathbf{x}^*} \right)^T \mathbf{e}_j = -2d y_i e^{-d\|\mathbf{x}_i - \mathbf{x}^*\|^2} (\mathbf{x}^* - \mathbf{x}_i)^T \mathbf{e}_j$$

$$\begin{aligned}
[\mathrm{H}]_{ij} &= \hat{\mathbf{x}}_i^T \frac{\partial^2 k(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i \partial \mathbf{x}_j} \hat{\mathbf{x}}_j = \hat{\mathbf{x}}_i^T \left[\frac{\partial}{\partial \mathbf{x}_i} \left\{ -2d e^{-d\|\mathbf{x}_i - \mathbf{x}_j\|^2} (\mathbf{x}_j - \mathbf{x}_i) \right\} \right] \hat{\mathbf{x}}_j \\
&= 2d e^{-d\|\mathbf{x}_i - \mathbf{x}_j\|^2} \left[\hat{\mathbf{x}}_i^T \hat{\mathbf{x}}_j - 2d \left\{ \hat{\mathbf{x}}_i^T (\mathbf{x}_i - \mathbf{x}_j) \right\} \left\{ (\mathbf{x}_i - \mathbf{x}_j)^T \hat{\mathbf{x}}_j \right\} \right].
\end{aligned}$$

Again, replacing $\mathbf{x}_j$ by $\mathbf{x}^*$,

$$[\mathrm{H}_*]_{ij} = \hat{\mathbf{x}}_i^T \frac{\partial^2 k(\mathbf{x}_i, \mathbf{x}^*)}{\partial \mathbf{x}_i \partial \mathbf{x}^*} \mathbf{e}_j = 2d e^{-d\|\mathbf{x}_i - \mathbf{x}^*\|^2} \left[\hat{\mathbf{x}}_i^T \mathbf{e}_j - 2d \left\{ \hat{\mathbf{x}}_i^T (\mathbf{x}_i - \mathbf{x}^*) \right\} \left\{ (\mathbf{x}_i - \mathbf{x}^*)^T \mathbf{e}_j \right\} \right].$$

Replacing $\mathbf{x}_i$ also by $\mathbf{x}^*$,

$$[\mathrm{H}_{**}]_{ij} = \mathbf{e}_i^T \frac{\partial^2 k(\mathbf{x}^*, \mathbf{x}^*)}{\partial \mathbf{x}^* \partial \mathbf{x}^*} \mathbf{e}_j = 2d \left(\mathbf{e}_i^T \mathbf{e}_j\right).$$

Polynomial Kernel

$$[\mathrm{K}]_{ij} = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) = y_i y_j (\mathbf{x}_i^T \mathbf{x}_j + 1)^d$$
$$[\mathrm{G}]_{ij} = y_i \left(\frac{\partial k(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_j}\right)^T \hat{\mathbf{x}}_j = y_i d (\mathbf{x}_i^T \mathbf{x}_j + 1)^{d-1} \mathbf{x}_i^T \hat{\mathbf{x}}_j.$$

Replacing $\mathbf{x}_j$ by $\mathbf{x}_*$ in the above equation we get

$$[\mathrm{G}_*]_{ij} = y_i \left(\frac{\partial k(\mathbf{x}_i, \mathbf{x}_*)}{\partial \mathbf{x}_*}\right)^T \mathbf{e}_j = y_i d \left(\mathbf{x}_i^T \mathbf{x}_* + 1\right)^{d-1} \mathbf{x}_i^T \mathbf{e}_j.$$

$$\begin{aligned}
[\mathrm{H}]_{ij} =& \hat{\mathbf{x}}_i^T \frac{\partial^2 k(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i \partial \mathbf{x}_j} \hat{\mathbf{x}}_j \\
&= \hat{\mathbf{x}}_i^T \left[\frac{\partial}{\partial \mathbf{x}_i} \left\{ d (\mathbf{x}_i^T \mathbf{x}_j + 1)^{d-1} \mathbf{x}_i \right\}\right] \hat{\mathbf{x}}_j \\
&= \hat{\mathbf{x}}_i^T \left[d (\mathbf{x}_i^T \mathbf{x}_j + 1)^{d-1} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_i} + \mathbf{x}_i \frac{\partial}{\partial \mathbf{x}_i} \left\{ d (\mathbf{x}_i^T \mathbf{x}_j + 1)^{d-1} \right\}\right] \hat{\mathbf{x}}_j \\
&= \hat{\mathbf{x}}_i^T \left[d (\mathbf{x}_i^T \mathbf{x}_j + 1)^{d-1} \mathrm{I}_N + d(d-1) (\mathbf{x}_i^T \mathbf{x}_j + 1)^{d-2} \mathbf{x}_i \mathbf{x}_j^T \right] \hat{\mathbf{x}}_j \\
&= d (\mathbf{x}_i^T \mathbf{x}_j + 1)^{d-2} \left[(\mathbf{x}_i^T \mathbf{x}_j + 1) \hat{\mathbf{x}}_i^T \hat{\mathbf{x}}_j + (d-1) \left(\hat{\mathbf{x}}_i^T \mathbf{x}_i\right) \left(\mathbf{x}_j^T \hat{\mathbf{x}}_j\right) \right].
\end{aligned}$$

Again, replacing $\mathbf{x}_j$ by $\mathbf{x}_*$,

$$\begin{aligned}
[\mathrm{H}_*]_{ij} &= \hat{\mathbf{x}}_i^T \frac{\partial^2 k(\mathbf{x}_i, \mathbf{x}_*)}{\partial \mathbf{x}_i \partial \mathbf{x}_*} \mathbf{e}_j \\
&= d (\mathbf{x}_i^T \mathbf{x}_* + 1)^{d-2} \left[(\mathbf{x}_i^T \mathbf{x}_* + 1) \hat{\mathbf{x}}_i^T \mathbf{e}_j + (d-1) \left(\hat{\mathbf{x}}_i^T \mathbf{x}_i\right) \left(\mathbf{x}_*^T \mathbf{e}_j\right) \right].
\end{aligned}$$

Replacing $\mathbf{x}_i$ also by $\mathbf{x}_*$,

$$\begin{aligned}
[\mathrm{H}_{**}]_{ij} &= \mathbf{e}_i^T \frac{\partial^2 k(\mathbf{x}_*, \mathbf{x}_*)}{\partial \mathbf{x}_*^2} \mathbf{e}_j \\
&= d (\mathbf{x}_*^T \mathbf{x}_* + 1)^{d-2} \left[(\mathbf{x}_*^T \mathbf{x}_* + 1) \mathbf{e}_i^T \mathbf{e}_j + (d-1) \left(\mathbf{e}_i^T \mathbf{x}_*\right) \left(\mathbf{x}_*^T \mathbf{e}_j\right) \right].
\end{aligned}$$

References

1. Chiang, H., Chu, C.: A systematic search method for obtaining multiple local optimal solutions of nonlinear programming problems. IEEE Trans. Circ. Syst. I Fundam. Theory Appl. **43**(2), 99–109 (1996)
2. Dixon, K., Khosla, P.: Trajectory representation using sequenced linear dynamical systems. In: Proceedings of 2004 IEEE International Conference on Robotics and Automation, 2004 (ICRA'04), vol. 4, pp. 3925–3930. IEEE (2004)
3. Ellekilde, L., Christensen, H.: Control of mobile manipulator using the dynamical systems approach. In: Proceedings of 2009 IEEE International Conference on Robotics and Automation, 2009 (ICRA'09), pp. 1370–1376. IEEE (2009)
4. Fuchs, A., Haken, H.: Pattern recognition and associative memory as dynamical processes in a synergetic system. I. Translational invariance, selective attention, and decomposition ofscenes. Biol. Cybern. **60**, 17–22 (1988). http://dl.acm.org/citation.cfm?id=56852.56854
5. Hoffmann, H.: Target switching in curved human arm movements is predicted by changing a single control parameter. Exp. Brain Res. **208**(1), 73–87 (2011)
6. Jaeger, H., Lukosevicius, M., Popovici, D., Siewert, U.: Optimization and applications of echo state networks with leaky-integrator neurons. Neural Netw. **20**(3), 335–352 (2007)
7. Khansari-Zadeh, S.M., Billard, A.: Learning stable non-linear dynamical systems with Gaussian mixture models. IEEE Trans. Robot. **27**(5), 943–957 (2011). http://lasa.epfl.ch/khansari
8. Lee, J.: Dynamic gradient approaches to compute the closest unstable equilibrium point for stability region estimate and their computational limitations. IEEE Trans. Automat. Contr. **48**(2), 321–324 (2003)
9. Michel, A., Farrell, J.: Associative memories via artificial neural networks. IEEE Contr. Syst. Mag. **10**(3), 6–17 (1990). doi:10.1109/37.55118
10. Pastor, P., Hoffmann, H., Asfour, T., Schaal, S.: Learning and generalization of motor skills by learning from demonstration. In: Proceedings of 2009 IEEE International Conference on Robotics and Automation, 2009 (ICRA '09), pp. 763–768 (2009). doi:10.1109/ROBOT.2009.5152385
11. Rasmussen, C.: Gaussian processes in machine learning. In: Advanced Lectures on Machine Learning, pp. 63–71. Springer, Berlin (2004)
12. Reimann, H., Iossifidis, I., Schöner, G.: Autonomous movement generation for manipulators with multiple simultaneous constraints using the attractor dynamics approach. In: Proceedings of 2011 IEEE International Conference on Robotics and Automation, 2011 (ICRA), pp. 5470–5477. IEEE (2011)
13. Schaal, S., Atkeson, C., Vijayakumar, S.: Scalable techniques from nonparametric statistics for real time robot learning. Appl. Intell. **17**(1), 49–60 (2002)
14. Schölkopf, B., Smola, A.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2001)
15. Schöner, G., Dose, M.: A dynamical systems approach to task-level system integration used to plan and control autonomous vehicle motion. Robot. Auton. Syst. **10**(4), 253–267 (1992)
16. Schöner, G., Dose, M., Engels, C.: Dynamics of behavior: theory and applications for autonomous robot architectures. Robot. Auton. Syst. **16**(2), 213–245 (1995)
17. Shukla, A., Billard, A.: Coupled dynamical system based armhand grasping model for learning fast adaptation strategies. Robot. Auton. Syst. **60**(3), 424–440 (2012). doi:10.1016/j.robot.2011.07.023. http://www.sciencedirect.com/science/article/pii/S0921889011001576