6

LEARNING NONLINEAR DYNAMICAL SYSTEMS USING THE EXPECTATION–MAXIMIZATION ALGORITHM

Sam Roweis and Zoubin Ghahramani

Gatsby Computational Neuroscience Unit, University College London, London U.K.

(zoubin@gatsby.ucl.ac.uk)

6.1 LEARNING STOCHASTIC NONLINEAR DYNAMICS

Since the advent of cybernetics, dynamical systems have been an important modeling tool in fields ranging from engineering to the physical and social sciences. Most realistic dynamical systems models have two essential features. First, they are stochastic – the observed outputs are a noisy function of the inputs, and the dynamics itself may be driven by some unobserved noise process. Second, they can be characterized by some finite-dimensional internal state that, while not directly observable, summarizes at any time all information about the past behavior of the process relevant to predicting its future evolution.

From a modeling standpoint, stochasticity is essential to allow a model with a few fixed parameters to generate a rich variety of time-series outputs.[1] Explicitly modeling the internal state makes it possible to decouple the internal dynamics from the observation process. For example, to model a sequence of video images of a balloon floating in the wind, it would be computationally very costly to directly predict the array of camera pixel intensities from a sequence of arrays of previous pixel intensities. It seems much more sensible to attempt to infer the true state of the balloon (its position, velocity, and orientation) and decouple the process that governs the balloon dynamics from the observation process that maps the actual balloon state to an array of measured pixel intensities.

Often we are able to write down equations governing these dynamical systems directly, based on prior knowledge of the problem structure and the sources of noise – for example, from the physics of the situation. In such cases, we may want to infer the hidden state of the system from a sequence of observations of the system's inputs and outputs. Solving this *inference* or *state-estimation* problem is essential for tasks such as tracking or the design of state-feedback controllers, and there exist well-known algorithms for this.

However, in many cases, the exact parameter values, or even the gross structure of the dynamical system itself, may be unknown. In such cases, the dynamics of the system have to be *learned* or *identified* from sequences of observations only. Learning may be a necessary precursor if the ultimate goal is effective state inference. But learning nonlinear state-based models is also useful in its own right, even when we are not explicitly interested in the internal states of the model, for tasks such as prediction (extrapolation), time-series classification, outlier detection, and filling-in of missing observations (imputation). This chapter addresses the problem of learning time-series models when the internal state is hidden. Below, we briefly review the two fundamental algorithms that form the basis of our learning procedure. In section 6.2, we introduce our algorithm

[1]There are, of course, completely deterministic but *chaotic* systems with this property. If we separate the noise processes in our models from the deterministic portions of the dynamics and observations, we can think of the noises as another deterministic (but highly chaotic) system that depends on initial conditions and exogenous inputs that we do not know. Indeed, when we run simulations using a psuedo-random-number generator started with a particular seed, this is precisely what we are doing.

and derive its learning rules. Section 6.3 presents results of using the algorithm to identify nonlinear dynamical systems. Finally, we present some conclusions and potential extensions to the algorithm in Sections 6.4 and 6.5.

6.1.1 State Inference and Model Learning

Two remarkable algorithms from the 1960s – one developed in engineering and the other in statistics – form the basis of modern techniques in state estimation and model learning. The Kalman filter, introduced by Kalman and Bucy in 1961 [1], was developed in a setting where the physical model of the dynamical system of interest was readily available; its goal is optimal state estimation in systems with known parameters. The expectation–maximization (EM) algorithm, pioneered by Baum and colleagues [2] and later generalized and named by Dempster et al. [3], was developed to learn parameters of statistical models in the presence of incomplete data or hidden variables.

In this chapter, we bring together these two algorithms in order to learn the dynamics of stochastic nonlinear systems with hidden states. Our goal is twofold: both to develop a method for identifying the dynamics of nonlinear systems whose hidden states we wish to infer, and to develop a general nonlinear time-series modeling tool. We examine inference and learning in discrete-time[2] stochastic nonlinear dynamical systems with hidden states x_k, external inputs u_k, and noisy outputs y_k. (All lower-case characters (except indices) denote vectors. Matrices are represented by upper-case characters.) The systems are parametrized by a set of tunable matrices, vectors, and scalars, which we shall collectively denote as θ. The inputs, outputs, and states are related to each other by

$$x_{k+1} = f(x_k, u_k) + w_k, \tag{6.1a}$$

$$y_k = g(x_k, u_k) + v_k, \tag{6.1b}$$

[2]Continuous-time dynamical systems (in which *derivatives* are specified as functions of the current state and inputs) can be converted into discrete-time systems by sampling their outputs and using "zero-order holds" on their inputs. In particular, for a continuous-time linear system $\dot{x}(t) = A_c x(t) + B_c u(t)$ sampled at interval τ, the corresponding dynamics and input driving matrices so that $x_{k+1} = Ax_k + Bu_k$ are $A = \sum_{k=0}^{\infty} A_c^k \tau^k / k! = \exp(A_c \tau)$ and $B = A_c^{-1}(A - I)B_c$.

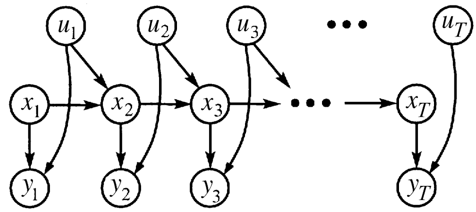

Figure 6.1 A probabilistic graphical model for stochastic dynamical systems with hidden states x_k, inputs u_k, and observables y_k.

where w_k and v_k are zero-mean Gaussian noise processes. The state vector x evolves according to a nonlinear but stationary Markov dynamics[3] driven by the inputs u and by the noise source w. The outputs y are nonlinear, noisy but stationary and instantaneous functions of the current state and current input. The vector-valued nonlinearities f and g are assumed to be differentiable, but otherwise arbitrary. The goal is to develop an algorithm that can be used to model the probability density of output sequences (or the conditional density of outputs given inputs) using only a finite number of example time series. The crux of the problem is that both the hidden state trajectory and the parameters are unknown.

Models of this kind have been examined for decades in systems and control engineering. They can also be viewed within the framework of *probabilistic graphical models*, which use graph theory to represent the conditional dependencies between a set of variables [4, 5]. A probabilistic graphical model has a node for each (possibly vector-valued) random variable, with directed arcs representing stochastic dependences. Absent connections indicate conditional independence. In particular, nodes are conditionally independent from their non-descendents, given their parents – where parents, children, descendents, etc, are defined with respect to the directionality of the arcs (i.e., arcs go from parent to child). We can capture the dependences in Eqs. (6.1*a*,*b*) compactly by drawing the graphical model shown in Figure 6.1.

One of the appealing features of probabilistic graphical models is that they explicitly diagram the mechanism that we assume generated the data. This *generative model* starts by picking randomly the values of the nodes that have no parents. It then picks randomly the values of their children

[3]Stationarity means here that neither f nor the covariance of the noise process w_k, depend on time; that is, the dynamics are *time-invariant*. Markov refers to the fact that given the current state, the next state does not depend on the past history of the states.

given the parents' values, and so on. The random choices for each child given its parents are made according to some assumed noise model. The combination of the graphical model and the assumed noise model at each node fully specify a probability distribution over all variables in the model.

Graphical models have helped clarify the relationship between dynamical systems and other probabilistic models such as hidden Markov models and factor analysis [6]. Graphical models have also made it possible to develop probabilistic inference algorithms that are vastly more general than the Kalman filter.

If we knew the parameters, the operation of interest would be to infer the hidden state sequence. The uncertainty in this sequence would be encoded by computing the posterior distributions of the hidden state variables given the sequence of observations. The Kalman filter (reviewed in Chapter 1) provides a solution to this problem in the case where f and g are linear. If, on the other hand, we had access to the hidden state trajectories as well as to the observables, then the problem would be one of model-fitting, i.e. estimating the parameters of f and g and the noise covariances. Given observations of the (no longer hidden) states and outputs, f and g can be obtained as the solution to a possibly nonlinear regression problem, and the noise covariances can be obtained from the residuals of the regression. How should we proceed when *both* the system model and the hidden states are unknown?

The classical approach to solving this problem is to treat the parameters θ as "extra" hidden variables, and to apply an extended Kalman filtering (EKF) algorithm (see Chapter 1) to the nonlinear system with the state vector augmented by the parameters [7, 8]. For stationary models, the dynamics of the parameter portion of this extended state vector are set to the identity function. The approach can be made inherently on-line, which may be important in certain applications. Furthermore, it provides an estimate of the covariance of the parameters at each time step. Finally, its objective, probabilistically speaking, is to find an optimum in the joint space of parameters and hidden state sequences.

In contrast, the algorithm we present is a batch algorithm (although, as we discuss in Section 6.4.2, online extensions are possible), and does not attempt to estimate the covariance of the parameters. Like other instances of the EM algorithm, which we describe below, its goal is to integrate over the uncertain estimates of the unknown hidden states and optimize the resulting marginal likelihood of the parameters given the observed data. An extended Kalman smoother (EKS) is used to estimate the approximate state distribution in the E-step, and a radial basis function (RBF) network [9, 10] is used for nonlinear regression in the M-step. It is important not to confuse this use of the extended Kalman algorithm, namely, to estimate just the hidden state as part of the E-step of EM, with the use that we described in the previous paragraph, namely to simultaneously estimate parameters and hidden states.

6.1.2 The Kalman Filter

Linear dynamical systems with additive white Gaussian noises are the most basic models to examine when considering the state-estimation problem, because they admit exact and efficient inference. (Here, and in what follows, we call a system linear if both the state evolution function and the state-to-output observation function are linear, and nonlinear otherwise.) The linear dynamics and observation processes correspond to matrix operations, which we denote by A, B and C, D, respectively, giving the classic state-space formulation of input-driven linear dynamical systems:

$$x_{k+1} = Ax_k + Bu_k + w_k, \tag{6.2a}$$

$$y_k = Cx_k + Du_k + v_k. \tag{6.2b}$$

The Gaussian noise vectors w and v have zero mean and covariances Q and R respectively. If the prior probability distribution $p(x_1)$ over initial states is taken to be Gaussian, then the joint probabilities of all states and outputs at future times are also Gaussian, since the Gaussian distribution is closed under the linear operations applied by state evolution and output mapping and under the convolution applied by additive Gaussian noise. Thus, all distributions over hidden state variables are fully described by their means and covariance matrices. The algorithm for exactly computing the posterior mean and covariance for x_k given some sequence of observations consists of two parts: a forward recursion, which uses the observations from y_1 to y_k, known as the *Kalman filter* [11], and a backward recursion, which uses the observations from y_T to y_{k+1}. The combined forward and backward recursions are known as the Kalman or Rauch–Tung–Streibel (RTS) *smoother* [12]. These algorithms are reviewed in detail in Chapter 1.

There are three key insights to understanding the Kalman filter. The first is that the Kalman filter is simply a method for implementing Bayes' rule. Consider the very general setting where we have a prior $p(x)$ on some state variable and an observation model $p(y|x)$ for the noisy outputs given the state. Bayes' rule gives us the state-inference procedure:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{Z}, \tag{6.3a}$$

$$Z = p(y) = \int_x p(y|x)p(x)\,dx, \tag{6.3b}$$

where the normalizer Z is the unconditional density of the observation. All we need to do in order to convert our prior on the state into a posterior is to multiply by the likelihood from the observation equation, and then renormalize.

The second insight is that there is no need to invert the output or dynamics functions, as long as we work with easily normalizable distributions over hidden states. We see this by applying Bayes' rule to the linear Gaussian case for a single time step.[4] We start with a Gaussian belief $\mathcal{N}(x_{k-1}, V_{k-1})$ on the current hidden state, use the dynamics to convert this to a prior $\mathcal{N}(x^+, V^+)$ on the next state, and then condition on the observation to convert this prior into a posterior $\mathcal{N}(x_k, V_k)$. This gives the classic Kalman filtering equations:

$$p(x_{k-1}) = \mathcal{N}(x^+, V^+), \tag{6.4a}$$

$$x^+ = Ax_{k-1}, \qquad V^+ = AV_{k-1}A^\top + Q, \tag{6.4b}$$

$$p(y_k|x_k) = \mathcal{N}(Cx_k, R), \tag{6.4c}$$

$$p(x_k|y_k) = \mathcal{N}(x_k, V_k), \tag{6.4d}$$

$$x_k = x^+ + K(y_k - Cx^+), \qquad V_k = (I - KC)V^+, \tag{6.4e}$$

$$K = V^+C^\top(CV^+C^\top + R)^{-1}. \tag{6.4f}$$

The posterior is again Gaussian and analytically tractable. Notice that neither the dynamics matrix A nor the observation matrix C needed to be inverted.

The third insight is that the state-estimation procedures can be implemented recursively. The posterior from the previous time step is run through the dynamics model and becomes our prior for the current time step. We then convert this prior into a new posterior by using the current observation.

[4]Some notation: A multivariate normal (Gaussian) distribution with mean μ and covariance matrix Σ is written as $\mathcal{N}(\mu, \Sigma)$. The same Gaussian evaluated at the point z is denoted by $\mathcal{N}(\mu, \Sigma)|_z$. The determinant of a matrix is denoted by $|A|$ and matrix inversion by A^{-1}. The symbol $\sim$ means "distributed according to."

For the general case of a nonlinear system with non-Gaussian noise, state estimation is much more complex. In particular, mapping through arbitrary nonlinearities f and g can result in arbitrary state distributions, and the integrals required for Bayes' rule can become intractable. Several methods have been proposed to overcome this intractability, each providing a distinct approximate solution to the inference problem. Assuming f and g are differentiable and the noise is Gaussian, one approach is to locally linearize the nonlinear system about the current state estimate so that applying the Kalman filter to the linearized system the approximate state distribution remains Gaussian. Such algorithms are known as *extended* Kalman filters (EKF) [13, 14]. The EKF has been used both in the classical setting of state estimation for nonlinear dynamical systems and also as a basis for on-line learning algorithms for feedforward neural networks [15] and radial basis function networks [16, 17]. For more details, see Chapter 2.

State inference in nonlinear systems can also be achieved by propagating a set of random samples in state space through f and g, while at each time step re-weighting them using the likelihood $p(y|x)$. We shall refer to algorithms that use this general strategy as *particle filters* [18], although variants of this sampling approach are known as sequential importance sampling, bootstrap filters [19], Monte Carlo filters [20], condensation [21], and dynamic mixture models [22, 23]. A recent survey of these methods is provided in [24]. A third approximate state-inference method, known as the *unscented* filter [25–27], deterministically chooses a set of balanced points and propagates them through the nonlinearities in order to recursively approximate a Gaussian state distribution; for more details, see Chapter 7. Finally, there are algorithms for approximate inference and learning based on mean field theory and variational methods [28, 29]. Although we have chosen to make local linearization (EKS) the basis of our algorithms below, it is possible to formulate the same learning algorithms using any approximate inference method (e.g., the unscented filter).

6.1.3 The EM Algorithm

The EM or *expectation–maximization* algorithm [3, 30] is a widely applicable iterative parameter re-estimation procedure. The objective of the EM algorithm is to maximize the likelihood of the observed data $P(Y|\theta)$ in the presence of hidden[5] variables X. (We shall denote the entire

[5]Hidden variables are often also called *latent variables*; we shall use both terms. They can also be thought of as missing data for the problem or as auxiliary parameters of the model.

sequence of observed data by $Y = \{y_1, \ldots, y_\tau\}$, observed inputs by $U = \{u_1, \ldots, u_T\}$, the sequence of hidden variables by $X = \{x_1, \ldots, x_\tau\}$, and the parameters of the model by θ.) Maximizing the likelihood as a function of θ is equivalent to maximizing the log-likelihood:

$$\mathcal{L}(\theta) = \log P(Y|U, \theta) = \log \int_X P(X, Y|U, \theta)\, dX. \tag{6.5}$$

Using *any* distribution $Q(X)$ over the hidden variables, we can obtain a lower bound on $\mathcal{L}$:

$$\begin{aligned}
\log \int_X P(Y, X|U, \theta)\, dX &= \log \int_X Q(X) \frac{P(X, Y|U, \theta)}{Q(X)}\, dX && (6.6a)\\
&\geq \int_X Q(X) \log \frac{P(X, Y|U, \theta)}{Q(X)}\, dX && (6.6b)\\
&= \int_X Q(X) \log P(X, Y|U, \theta)\, dX \\
&\quad - \int_X Q(X) \log Q(X)\, dX && (6.6c)\\
&= \mathcal{F}(Q, \theta), && (6.6d)
\end{aligned}$$

where the middle inequality (6.6*b*) is known as Jensen's inequality and can be proved using the concavity of the log function. If we define the *energy* of a global configuration (X, Y) to be $-\log P(X, Y|U, \theta)$, then the lower bound $\mathcal{F}(Q, \theta) \leq \mathcal{L}(\theta)$ is the negative of a quantity known in statistical physics as the *free energy*: the expected energy under Q minus the entropy of Q [31]. The EM algorithm alternates between maximizing $\mathcal{F}$ with respect to the *distribution* Q and the *parameters* θ, respectively, holding the other fixed. Starting from some initial parameters θ_0 we alternately apply:

$$\textbf{E-step:} \quad Q_{k+1} \leftarrow \arg\max_Q \; \mathcal{F}(Q, \theta_k), \tag{6.7a}$$

$$\textbf{M-step:} \quad \theta_{k+1} \leftarrow \arg\max_\theta \mathcal{F}(Q_{k+1}, \theta). \tag{6.7b}$$

It is easy to show that the maximum in the E-step results when Q is exactly the conditional distribution of X, $Q^*_{k+1}(X) = P(X|Y, U, \theta_k)$, at which point the bound becomes an equality: $\mathcal{F}(Q^*_{k+1}, \theta_k) = \mathcal{L}(\theta_k)$. The maximum in the M-step is obtained by maximizing the first term in (6.6*c*), since the entropy of Q does not depend on θ:

$$\textbf{M-step:} \qquad \theta^*_{k+1} \leftarrow \arg\max_{\theta} \int_X P(X|Y, U, \theta_k) \log P(X, Y|U, \theta)\, dX. \tag{6.8}$$

This is the expression most often associated with the EM algorithm, but it obscures the elegant interpretation [31] of EM as coordinate ascent in $\mathcal{F}$ (see Fig. 6.2). Since $\mathcal{F} = \mathcal{L}$ at the beginning of each M-step, and since the E-step does not change θ, we are guaranteed not to decrease the likelihood after each combined EM step. (While this is obviously true of "complete" EM algorithms as described above, it may also be true for "incomplete" or "sparse" variants in which approximations are used during the E- and/or M-steps so long as $\mathcal{F}$ always goes up; see also the earlier work in [32].) For example, this can take the form of a gradient M- step algorithm (where we increase $P(Y|\theta)$ with respect to θ but do not strictly maximize it), or any E-step which improves the bound $\mathcal{F}$ without saturating it [31].)

In dynamical systems with hidden states, the E-step corresponds exactly to solving the smoothing problem: estimating the hidden state trajectory given both the observations/inputs and the parameter values. The M-step involves system identification using the state estimates from the smoother. Therefore, at the heart of the EM learning procedure is the following idea: use the solutions to the filtering/smoothing problem to estimate the unknown hidden states given the observations and the current

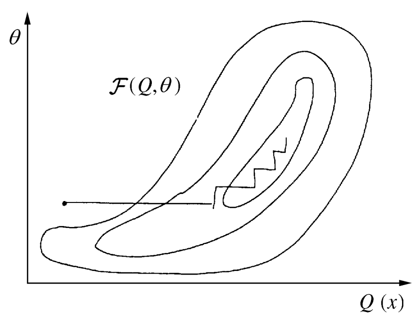

Figure 6.2 The EM algorithm can be thought of as coordinate ascent in the functional $\mathcal{F}(Q(X), \theta)$ (see text). The E-step maximizes $\mathcal{F}$ with respect to $Q(X)$ given fixed θ (horizontal moves), while the M-step maximizes $\mathcal{F}$ with respect to θ given fixed $Q(X)$ (vertical moves).

model parameters. Then use this fictitious complete data to solve for new model parameters. Given the estimated states obtained from the inference algorithm, it is usually easy to solve for new parameters. For example, when working with linear Gaussian models, this typically involves minimizing quadratic forms, which can be done with linear regression. This process is repeated, using these new model parameters to infer the hidden states again, and so on. Keep in mind that our goal is to maximize the log-likelihood (6.5) (or equivalently maximize the total likelihood) of the observed data with respect to the model parameters. This means integrating (or summing) over all the ways in which the model could have produced the data (i.e., hidden state sequences). As a consequence of using the EM algorithm to do this maximization, we find ourselves needing to compute (and maximize) the *expected* log-likelihood of the *joint* data (6.8), where the expectation is taken over the distribution of hidden values predicted by the current model parameters and the observations.

In the past, the EM algorithm has been applied to learning linear dynamical systems in specific cases, such as "multiple-indicator multiple-cause" (MIMC) models with a single latent variable [33] or state-space models with the observation matrix known [34]), as well as more generally [35]. This chapter applies the EM algorithm to learning nonlinear dynamical systems, and is an extension of our earlier work [36]. Since then, there has been similar work applying EM to nonlinear dynamical systems [37, 38]. Whereas other work uses sampling for the E-step and gradient M-steps, our algorithm uses the RBF networks to obtain a computationally efficient and exact M-step.

The EM algorithm has four important advantages over classical approaches. First, it provides a straightforward and principled method for handing missing inputs or outputs. (Indeed this was the original motivation for Shumway and Stoffer's application of the EM algorithm to learning partially unknown linear dynamical systems [34].) Second, EM generalizes readily to more complex models with combinations of discrete and real-valued hidden variables. For example, one can formulate EM for a *mixture* of nonlinear dynamical systems [39, 40]. Third, whereas it is often very difficult to prove or analyze stability within the classical on-line approach, the EM algorithm is always attempting to maximize the likelihood, which acts as a Lyapunov function for stable learning. Fourth, the EM framework facilitates Bayesian extensions to learning – for example, through the use of variational approximations [29].

6.2 COMBINING EKS AND EM

In the next sections, we shall describe the basic components of our EM learning algorithm. For the expectation step of the algorithm, we infer an approximate conditional distribution of the hidden states using Extended Kalman Smoothing (Section 6.2.1). For the maximization step, we first discuss the general case (Section 6.2.2), and then describe the particular case where the nonlinearities are represented using Gaussian radial basis function (RBF) networks (Section 6.2.3). Since, as with all EM or likelihood ascent algorithms, our algorithm is not guaranteed to find the globally optimum solutions, good initialization is a key factor in practical success. We typically use a variant of factor analysis followed by estimation of a purely linear dynamical system as the starting point for training our nonlinear models (Section 6.2.4).

6.2.1 Extended Kalman smoothing (E-step)

Given a system described by Eqs. (6.1a,b), the E-step of an EM learning algorithm needs to infer the hidden states from a history of observed inputs and outputs. The quantities at the heart of this inference problem are two conditional densities

$$P(x_k|u_1, \ldots, u_T, y_1, \ldots, y_T), \qquad 1 \le k \le T, \tag{6.9}$$

$$P(x_k, x_{k+1}|u_1, \ldots, u_T, y_1, \ldots, y_T), \qquad 1 \le k \le T-1. \tag{6.10}$$

For nonlinear systems, these conditional densities are in general non-Gaussian, and can in fact be quite complex. For all but a very few nonlinear systems, exact inference equations cannot be written down in closed form. Furthermore, for many nonlinear systems of interest, exact inference is intractable (even numerically), meaning that, in principle, the amount of computation required grows exponentially in the length of the time series observed. The intuition behind all extended Kalman algorithms is that they approximate a stationary nonlinear dynamical system with a *non-stationary* (time-varying) but linear system. In particular, extended Kalman smoothing (EKS) simply applies regular Kalman smoothing to a local linearization of the nonlinear system. At every point $\tilde{x}$ in x space, the derivatives of the vector-valued functions f and g define the matrices,

$$A_{\tilde{x}} \equiv \left.\frac{\partial f}{\partial x}\right|_{x=\tilde{x}} \qquad \text{and} \qquad C_{\tilde{x}} \equiv \left.\frac{\partial g}{\partial x}\right|_{x=\tilde{x}},$$

respectively. The dynamics are linearized about $\hat{x}_k$, the mean of the current filtered (not smoothed) state estimate at time t. The output equation can be similarly linearized. These linearizations yield

$$x_{k+1} \approx f(\hat{x}_k, u_k) + A_{\hat{x}_k}(x_k - \hat{x}_k) + w, \tag{6.11}$$

$$y_k \approx g(\hat{x}_k, u_k) + C_{\hat{x}_k}(x_k - \hat{x}_k) + v. \tag{6.12}$$

If the noise distributions and the prior distribution of the hidden state at $k = 1$ are Gaussian, then, in this progressively linearized system, the conditional distribution of the hidden state at any time k given the history of inputs and outputs will also be Gaussian. Thus, Kalman smoothing can be used on the linearized system to infer this conditional distribution; this is illustrated in Figure 6.3.

Notice that although the algorithm performs *smoothing* (in other words, it takes into account all observations, including future ones, when inferring the state at any time), the linearization is only done in the forward direction. Why not re-linearize about the backwards estimates during the RTS recursions? While, in principle, this approach might give better results, it is difficult to implement in practice because it requires the dynamics functions to be uniquely invertible, which it often is not true.

Unlike the normal (linear) Kalman smoother, in the EKS, the error covariances for the state estimates and the Kalman gain matrices *do*

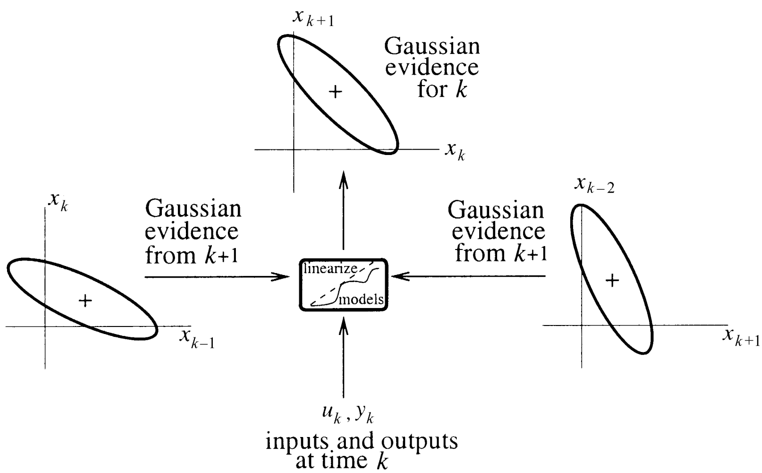

Figure 6.3 Illustration of the information used in extended Kalman smoothing (EKS), which infers the hidden state distribution during the E-step of our algorithm. The nonlinear model is linearized about the current state estimate at each time, and then Kalman smoothing is used on the linearized system to infer Gaussian state estimates.

depend on the observed data, not just on the time index t. Furthermore, it is no longer necessarily true that if the system is stationary, the Kalman gain will converge to a value that makes the smoother act as the optimal Wiener filter in the steady state.

6.2.2 Learning Model Parameters (M-step)

The M-step of our EM algorithm re-estimates the parameters of the model given the observed inputs, outputs, and the conditional distributions over the hidden states. For the model we have described, the parameters define the nonlinearities f and g, and the noise covariances Q and R (as well as the mean and covariance of the initial state, x_1).

Two complications can arise in the M-step. First, fully re-estimating f and g in each M-step may be computationally expensive. For example, if they are represented by neural network regressors, a single full M-step would be a lengthy training procedure using backpropagation, conjugate gradients, or some other optimization method. To avoid this, one could use partial M-steps that increase but do not maximize the expected log-likelihood (6.8) – for example, each consisting of one or a few gradient steps. However, this will in general make the fitting procedure much slower.

The second complication is that f and g have to be trained using the *uncertain* state-estimates output by the EKS algorithm. This makes it difficult to apply standard curve-fitting or regression techniques. Consider fitting f, which takes as inputs x_k and u_k and outputs x_{k+1}. For each t, the conditional density estimated by EKS is a full-covariance Gaussian in (x_k, x_{k+1}) space. So f has to be fit not to a set of data points but instead to a mixture of full-covariance Gaussians in input–output space (Gaussian "clouds" of data). Ideally, to follow the EM framework, this conditional density should be *integrated over* during the fitting process. Integrating over this type of data is nontrivial for almost any form of f. One simple but inefficient approach to bypass this problem is to draw a large sample from these Gaussian clouds of data and then fit f to these samples in the usual way. A similar situation occurs with the fitting of the output function g.

We present an alternative approach, which is to choose the form of the function approximator to make the integration easier. As we shall show, using Gaussian radial basis function (RBF) networks [9, 10] to model f and g allows us to do the integrals exactly and efficiently. With this choice of representation, both of the above complications vanish.

6.2.3 Fitting Radial Basis Functions to Gaussian Clouds

We shall present a general formulation of an RBF network from which it should be clear how to fit special forms for f and g. Consider the following nonlinear mapping from input vectors x and u to an output vector z:

$$z = \sum_{i=1}^{I} h_i \rho_i(x) + Ax + Bu + b + w, \tag{6.13}$$

where w is a zero-mean Gaussian noise variable with covariance Q, and ρ_i are scalar valved RBFs defined below. This general mapping can be used in several ways to represent dynamical systems, depending on which of the input to hidden to output mappings are assumed to be nonlinear. Three examples are: (1) representing f using (6.13) with the substitutions $x \leftarrow x_k$, $u \leftarrow u_k$, and $z \leftarrow x_{k+1}$; (2) representing f using $x \leftarrow (x_k, u_k)$, $u \leftarrow \emptyset$, and $z \leftarrow x_{k+1}$; and (3) representing g using the substitutions $x \leftarrow x_k$, $u \leftarrow u_k$, and $z \leftarrow y_k$. (Indeed, for different simulations, we shall use different forms.) The parameters are the I coefficients h_i of the RBFs; the matrices A and B multiplying inputs x and u, respectively; and an output bias vector b, and the noise covariance Q. Each RBF is assumed to be a Gaussian in x space, with center c_i and width given by the covariance matrix S_i:

$$\rho_i(x) = |2\pi S_i|^{-1/2} \exp[-\tfrac{1}{2}(x - c_i)^\top S_i^{-1}(x - c_i)], \tag{6.14}$$

where $|S_i|$ is the determinant of the matrix S_i. For now, we assume that the centers and widths of the RBFs are fixed, although we discuss learning their locations in Section 6.4.

The goal is to fit this RBF model to data (u, x, z). The complication is that the data set comes in the form of a mixture of Gaussian distributions. Here we show how to analytically integrate over this mixture distribution to fit the RBF model.

Assume the data set is

$$P(x, z, u) = \frac{1}{J} \sum_j \mathcal{N}_j(x, z)\delta(u - u_j). \tag{6.15}$$

That is, we observe samples from the u variables, each paired with a Gaussian "cloud" of data, $\mathcal{N}_j$, over (x, z). The Gaussian $\mathcal{N}_j$ has mean μ_j and covariance matrix C_j.

Let $\hat{z}_\theta(x, u) = \sum_{i=1}^{I} h_i \rho_i(x) + Ax + Bu + b$, where θ is the set of parameters. The log-likelihood of a single fully observed data point under the model would be

$$-\tfrac{1}{2}[z - \hat{z}_\theta(x, u)]^\top Q^{-1}[z - \hat{z}_\theta(x, u)] - \tfrac{1}{2}\ln|Q| + \text{const.}$$

Since the (x, z) values in the data set are uncertain, the maximum expected log-likelihood RBF fit to the mixture of Gaussian data is obtained by minimizing the following integrated quadratic form:

$$\min_{\theta,Q}\left\{\sum_j \int_x \int_z \mathcal{N}_j(x, z)[z - \hat{z}_\theta(x, u_j)]^\top Q^{-1}[z - \hat{z}_\theta(x, u_j)]\, dx\, dz + J\ln|Q|\right\}. \tag{6.16}$$

We rewrite this in a slightly different notation, using angular brackets $\langle\cdot\rangle_j$ to denote expectation over $\mathcal{N}_j$, and defining

$$\begin{aligned}\theta &\equiv [h_1, h_2, \ldots, h_I, A, B, b],\\ \Phi &\equiv [\rho_1(x), \rho_2(x), \ldots, \rho_I(x), x^\top, u^\top, 1]^\top.\end{aligned}$$

Then, the objective is written as

$$\min_{\theta,Q}\left\{\sum_j \langle (z - \theta\Phi)^\top Q^{-1}(z - \theta\Phi)\rangle_j + J\ln|Q|\right\}. \tag{6.17}$$

Taking derivatives with respect to θ, premultiplying by $-Q^{-1}$, and setting the result to zero gives the linear equations $\sum_j \langle (z - \theta\Phi)\Phi^T\rangle_j = 0$, which we can solve for θ and Q:

$$\hat{\theta} = \left(\sum_j \langle z\Phi^\top\rangle_j\right)\left(\sum_j \langle \Phi\Phi^\top\rangle_j\right)^{-1}, \quad \hat{Q} = \frac{1}{J}\left(\sum_j \langle zz^\top\rangle_j - \hat{\theta}\sum_j \langle \Phi z^\top\rangle_j\right). \tag{6.18}$$

In other words, given the expectations in the angular brackets, the optimal parameters can be solved for via a set of linear equations. In the Appendix, we show that these expectations can be computed analytically and efficiently, which means that we can take full and exact M-steps. The derivation is somewhat laborious, but the intuition is very simple: the

Gaussian RBFs multiply the Gaussian densities $\mathcal{N}_j$ to form new unnormalized Gaussians in (x, y) space. Expectations under these new Gaussians are easy to compute. This fitting algorithm is illustrated in Figure 6.4.

Note that among the four advantages we mentioned previously for the EM algorithm – ability to handle missing observations, generalizability to extensions of the basic model, Bayesian approximations, and guaranteed stability through a Lyapunov function – we have had to forgo one. There is no guarantee that extended Kalman smoothing increases the lower bound on the true likelihood, and therefore stability cannot be assured. In practice, the algorithm is rarely found to become unstable, and the approximation works well: in our experiments, the likelihoods increased monotonically and good density models were learned. Nonetheless, it may be desirable to derive guaranteed-stable algorithms for certain special cases using lower-bound preserving variational approximations [29] or other approaches that can provide such proofs.

The ability to fully integrate over uncertain state estimates provides practical benefits as well as being theoretically pleasing. We have compared fitting our RBF networks using only the means of the state estimates with performing the full integration as derived above. When using only the means, we found it necessary to introduce a ridge

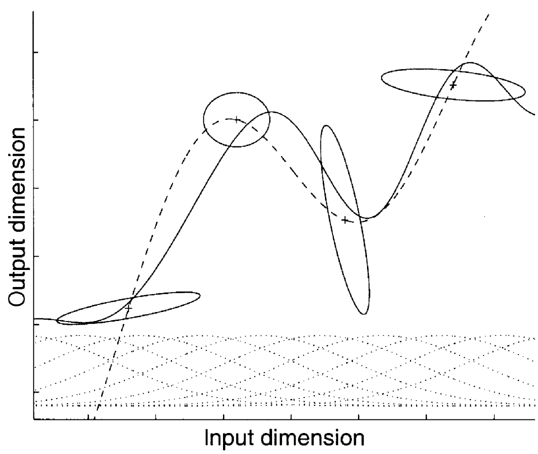

Figure 6.4 Illustration of the regression technique employed during the M-step. A fit to a mixture of Gaussian densities is required; if Gaussian RBF networks are used, then this fit can be solved analytically. The dashed line shows a regular RBF fit to the centers of the four Gaussian densities, while the solid line shows the analytical RBF fit using the covariance information. The dotted lines below show the support of the RBF kernels.

regression (weight decay) parameter in the M-step to penalize the very large coefficients that would otherwise occur based on precise cancellations between inputs. Since the model is linear in the parameters, this ridge regression regularizer is like adding white noise to the radial basis outputs $\rho_i(x)$ (i.e., *after* the RBF kernels have been applied).[6] By linearization, this is approximately equivalent to Gaussian noise at the inputs x with a covariance determined by the derivatives of the RBFs at the input locations. The uncertain state estimates provide exactly this sort of noise, and thus automatically regularize the RBF fit in the M-step. This naturally avoids the need to introduce a penalty on large coefficients, and improves generalization.

6.2.4 Initialization of Models and Choosing Locations for RBF Kernels

The practical success of our algorithm depends on two design choices that need to be made at the beginning of the training procedure. The first is to judiciously select the placement of the RBF kernels in the representation of the state dynamics and/or output function. The second is to sensibly initialize the parameters of the model so that iterative improvement with the EM algorithm (which finds only local maxima of the likelihood function) finds a good solution.

In models with low-dimensional hidden states, placement of RBF kernel centers can be done by gridding the state space and placing one kernel on each grid point. Since the scaling of the state variables is given by the covariance matrix of the state dynamics noise w_k in Eq. (6.1a) which, without loss of generality, we have set to I, it is possible to determine both a suitable size for the gridding region over the state space, and a suitable scaling of the RBF kernels themselves. However, the number of kernels in such a grid increases exponentially with the grid dimension, so, for more than three or four state variables, gridding the state space is impractical. In these cases, we first use a simple initialization, such as a linear dynamical system, to infer the hidden states, and then place RBF kernels on a randomly chosen subset of the inferred state means.[7] We set the widths (variances) of the RBF kernels once we have

[6] Consider a simple scalar linear regression example $y_j = \theta z_j$, which can be solved by minimizing $\sum_j (y_j - \theta z_j)^2$. If each z_j has mean $\bar{z}_j$ and variance λ, the expected value of this cost function is $\sum_j (y_j - \theta \bar{z}_j)^2 + J\lambda\theta^2$, which is exactly ridge regression with λ controlling the amount of regularization.

[7] In order to properly cover the portions of the state space that are most frequently used, we require a minimum distance between RBF kernel centers. Thus, in practice, we reject centers that fall too close together.

the spacing of their centers by attempting to make neighboring kernels cross when their outputs are half of their peak value. This ensures that, with all the coefficients set approximately equal, the RBF network will have an almost "flat" output across the space.[8]

These heuristics can be used both for fixed assignments of centers and widths, and as initialization to an adaptive RBF placement procedure. In Section 6.4.1, we discuss techniques for adapting both the positions of the RBF centers and their widths during training of the model.

For systems with nonlinear dynamics but approximately linear output functions, we initialize using maximum-likelihood factor analysis (FA) trained on the collection of output observations (or conditional factor analysis for models with inputs). Factor analysis is a very simple model, which assumes that the output variables are generated by linearly combining a small number of independent Gaussian hidden state variables and then adding independent Gaussian noise to each output variable [6]. One can think of factor analysis as a special case of linear dynamical systems with Gaussian noise where the states are not related in time (i.e., $A = 0$). We used the weight matrix (called the loading matrix) learned by factor analysis to initialize the observation matrix C in the dynamical system. By doing time-independent inference through the factor analysis model, we can also obtain approximate estimates for the state at each time. These estimates can be used to initialize the nonlinear RBF regressor by fitting the estimates at one time step as a function of those at the previous time step. (We also sometimes do a few iterations of training using a purely linear dynamical system before initializing the nonlinear RBF network.) Since such systems are nonlinear flows embedded in linear manifolds, this initialization estimates the embedding manifold using a linear statistical technique (FA) and the flow using a nonlinear regression based on projections into the estimated manifold.

If the output function is nonlinear but the dynamics are approximately linear, then a mixture of factor analyzers (MFA) can be trained on the output observations [41, 42]. A mixture of factor analyzers is a model that assumes that the data were generated from several Gaussian clusters with differing means, with the covariance within each cluster being modeled by a factor analyzer. Systems with nonlinear output function but linear dynamics capture linear flows in a nonlinear embedding manifold, and

[8]One way to see this is to consider Gaussian RBFs in an n-dimensional grid (i.e., a square lattice), all with heights 1. The RBF centers define a hypercube, the distance between neighboring RBFs being $2d$, where d is chosen such that $e^{-d^2/(2\sigma^2)} = \frac{1}{2}$. At the centers of the hypercubes, there are 2^n contributions from neighboring Gaussians, each of which is a distance $\sqrt{n}d$, and so contributes $(\frac{1}{2})^n$ to the height. Therefore, the height at the interiors is approximately equal to the height at the corners.

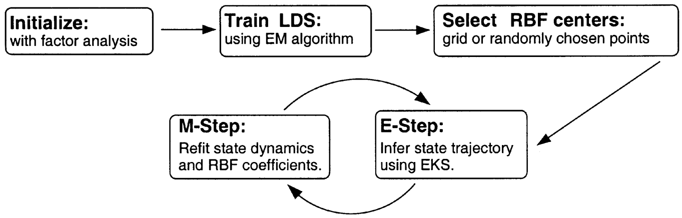

Figure 6.5 Summary of the main steps of the NLDS-EM algorithm.

the goal of the MFA initialization is to capture the nonlinear shape of the output manifold. Estimating the dynamics is difficult (since the hidden states of the individual analyzers in the mixture cannot be combined easily into a single internal state representation), but is still possible.[9] A summary of the algorithm including these initialization techniques is shown in Figure 6.5.

Ideally, Bayesian methods would be used to control the complexity of the model by estimating the internal state dimension and optimal number of RBF centers. However, in general, only approximate techniques such as cross-validation or variational approximations can be implemented in practice (see Section 6.4.4). Currently, we have set these complexity parameters either by hand or with cross-validation.

6.3 RESULTS

We tested how well our algorithm could learn the dynamics of a nonlinear system by observing only the system inputs and outputs. We investigated the behavior on simple one- and two-dimensional state-space problems whose nonlinear dynamics were known, as well as on a weather time-series problem involving real temperature data.

6.3.1 One- and Two-Dimensional Nonlinear State-Space Models

In order to be able to compare our algorithm's learned internal state representation with a ground truth state representation, we first tested it on

[9]As an approximate solution to the problem of getting a single hidden state from a MFA, we can use the following procedure: (1) Estimate the "similarity" between analyzer centers using average separation in time between data points for which they are active. (2) Use standard embedding techniques such as multidimensional scaling (MDS) [43] to place the MFA centers in a Euclidean space of dimension k. (3) Time-independent state inference for each observation now consists of the responsibility-weighted low-dimensional MFA centers, where the responsibilities are the posterior probabilities of each analyzer given the observation under the MFA.

synthetic data generated by nonlinear dynamics whose form was known. The systems we considered consisted of three inputs and four observables at each time, with either one or two hidden state variables. The relation of the state from one time step to the next was given by a variety of nonlinear functions followed by Gaussian noise. The outputs were a linear function of the state and inputs plus Gaussian noise. The inputs affected the state only through a linear driving function. The true and learned state transition functions for these systems, as well as sample outputs in response to Gaussian noise inputs and internal driving noise, are shown in Figures 6.6*c*,*d*, 6.7*c*, and 6.8*c*.

We initialized each nonlinear model with a linear dynamical model trained with EM, which, in turn, we initialized with a variant of factor analysis (see Section 6.2.4). The one-dimensional state-space models were given 11 RBFs in x space, which were uniformly spaced. (The range of maximum and minimum x values was automatically determined from the density of inferred points.) Two-dimensional state-space models were given 25 RBFs spaced in a 5×5 grid uniformly over the range of inferred

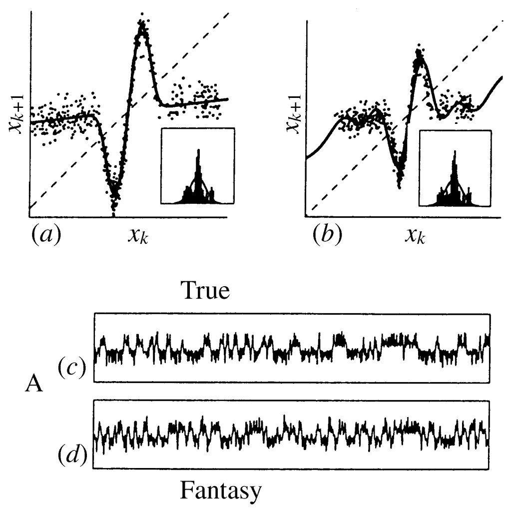

Figure 6.6 Example of fitting a system with nonlinear dynamics and linear observation function. The panels show the fitting of a nonlinear system with a one-dimensional hidden state and 4 noisy outputs driven by Gaussian noise inputs and internal state noise. (*a*) The true dynamics function (line) and states (dots) used to generate the training data (the inset is the histogram of internal states). (*b*) The learned dynamics function and states inferred on the training data (the inset is the histogram of inferred internal states). (*c*) The first component of the observable time series from the training data. (*d*) The first component of fantasy data generated from the learned model (on the same scale as *c*).

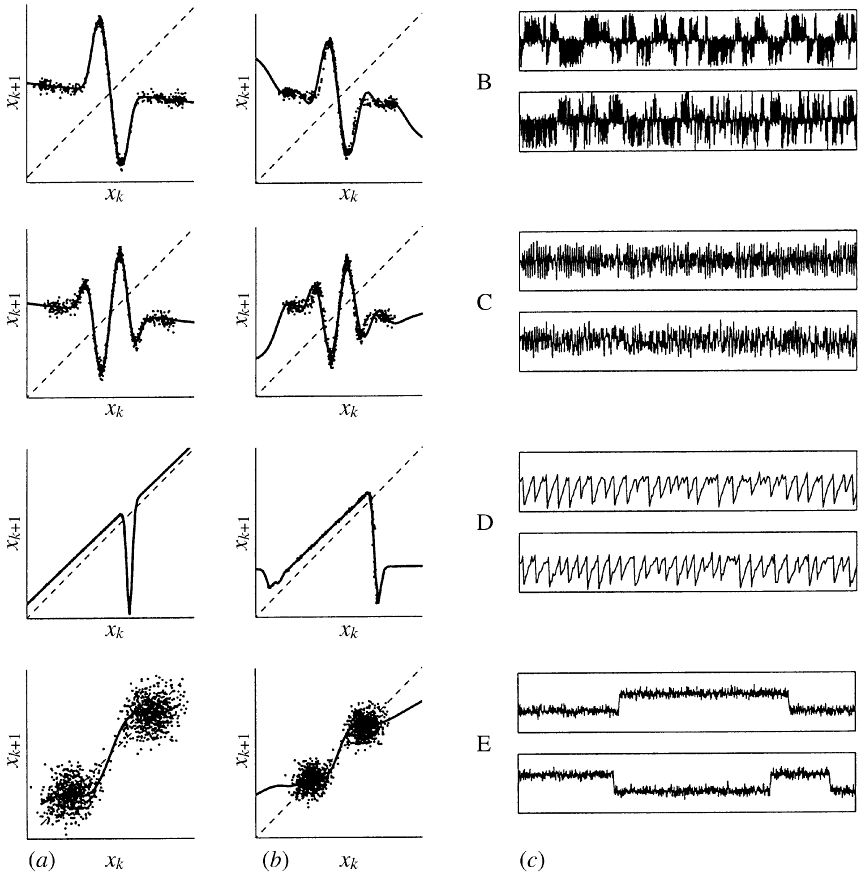

Figure 6.7 More examples of fitting systems with nonlinear dynamics and linear observation functions. Each of the five rows shows the fitting of a nonlinear system with a one-dimensional hidden state and four noisy outputs driven by Gaussian noise inputs and internal-state noise. (*a*) The true dynamics function (line) and states (dots) used to generate the training data. (*b*) The learned dynamics function and states inferred on the training data. (*c*) The first component of the observable time series: training data on the top and fantasy data generated from the learned model on the bottom. The nonlinear dynamics can produce quasi-periodic outputs in response to white driving noise.

states. After the initialization was over, the algorithm discovered the nonlinearities in the dynamics within less than 5 iterations of EM (see Figs. 6.6*a*,*b*, 6.7*a*,*b*, and 6.8*a*,*b*.

After training the models on input–output observations from the dynamics, we examined the learned internal state representation and

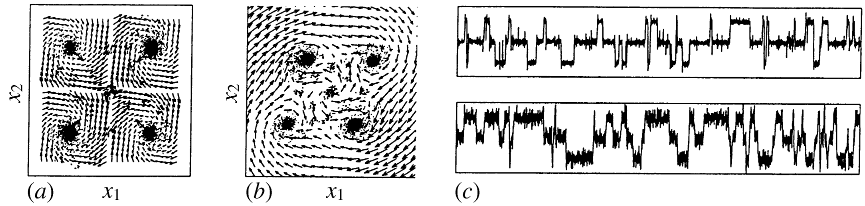

Figure 6.8 Multidimensional example of fitting a system with nonlinear dynamics and linear observation functions. The true system is piecewise-linear across the state space. The plots show the fitting of a nonlinear system with a two-dimensional hidden state and 4 noisy outputs driven by Gaussian noise inputs and internal state noise. (*a*) The true dynamics vector field (arrows) and states (dots) used to generate the training data. (*b*) The learned dynamics vector field and states inferred on the training data. (*c*) The first component of the observable time series: training data on the top and fantasy data generated from the learned model on the bottom.

compared it with the known structure of the generating system. As the figures show, the algorithm recovers the form of the nonlinear dynamics quite well. We are also able to generate "fantasy" data from the models once they have been learned by exciting them with Gaussian noise of similar variance to that applied during training. The resulting observation streams look qualitatively very similar to the time series from the true systems.

We can quantify this quality of fit by comparing the log-likelihood of the training sequences and novel test sequences under our nonlinear model with the likelihood under a basic linear dynamical system model or a static model such as factor analysis. Figure 6.9 presents this comparison. The nonlinear dynamical system had significantly superior likelihood on both training and test data for all the example systems. (Notice that for system **E**, the linear dynamical system is much better than factor analysis because of the strong hysteresis (mode-locking) in the system. Thus, the output at the previous time step is an excellent predictor of the current output.)

6.3.2 Weather Data

As an example of a real system with a nonlinear *output* function as well as important dynamics, we trained our model on records of the daily maximum and minimum temperatures in Melbourne, Australia, over the period 1981–1990.[10] We used a model with two internal state variables,

[10]This data is available on the world wide web from the Australian Bureau of Meteorology at http://www.bom.gov.au/climate.

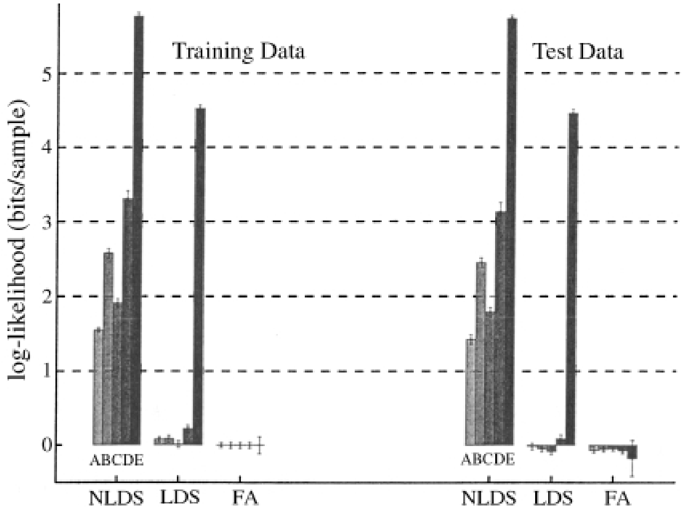

Figure 6.9 Differences in log-likelihood assigned by various models to training and test data from the systems in Figures 6.6 and 6.7. Each adjacent group of five bars shows the log-likelihood of the five examples (A–E) under factor analysis (FA), linear dynamical systems (LDS), and our nonlinear dynamical system model (NLDS). Results on training data appear on the left and results on test data on the right; taller bars represent better models. Log-likelihoods are offset so that FA on the training data is zero. Error bars represent the 68% quantile about the median across 100 repetitions of training or testing. For NLDS, the exact likelihood cannot be computed; what is shown is the pseudo-likelihood computed by EKS.

three outputs, and no inputs. During the training phase, the three outputs were the minimum and maximum daily temperature as well as a real valued output indicating the time of the year (month) in the range [0, 12]. The model was trained on 1500 days of temperature records, or just over four seasons. We tested on the remaining 2150 days by showing the model only the minimum and maximum daily temperatures and attempting to predict the time of year (month). The prediction was performed by using the EKS algorithm to do state inference given only the two available observation streams. Once state inference was performed, the learned output function of the model could be used to predict the time of year. This prediction problem inherently requires the use of information from previous and/or future times, since the static relationship between temperature and season is ambiguous during spring/fall. Figure 6.10 shows the results of this prediction after training; the algorithm has discovered a relationship between the hidden state and the observations that allows it to perform reasonable prediction for this task. Also shown

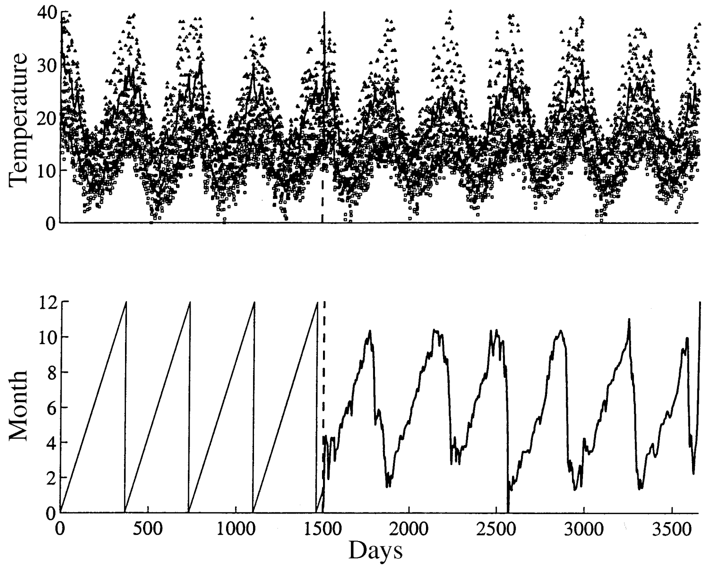

Figure 6.10 Model of maximum and minimum daily temperatures in Melbourne, Australia from 1981 to 1990. *Left of vertical line*: A system with two hidden states governed by linear dynamics and a non-linear output function was trained on observation vectors of a three-dimensional time series consisting of maximum and minimum temperatures for each day as well as the (real-valued) month of the year. Training points are shown as triangles (maximum temperature), squares (minimum temperature) and a solid line (sawtooth wave below). *Right of vertical line*: After training, the system can infer its internal state from only the temperature observations. Having inferred its internal state it can predict the month of the year as a missing output (line below). The solid lines in the upper plots show the model's prediction of minimum and maximum temperature given the inferred state at the time.

are the model predictions of minimum and maximum temperatures given the inferred state.

Although not explicitly part of the generative model, the learned system implicitly parameterizes a relationship between time of year and temperature. We can discover this relationship by evaluating the nonlinear output function at many points in the state space. Each evaluation yields a triple of month, minimum temperature and maximum temperature. These triples can then be plotted against each other as in Figure 6.11 to show that the model has discovered Melbourne's seasonal temperature variations.

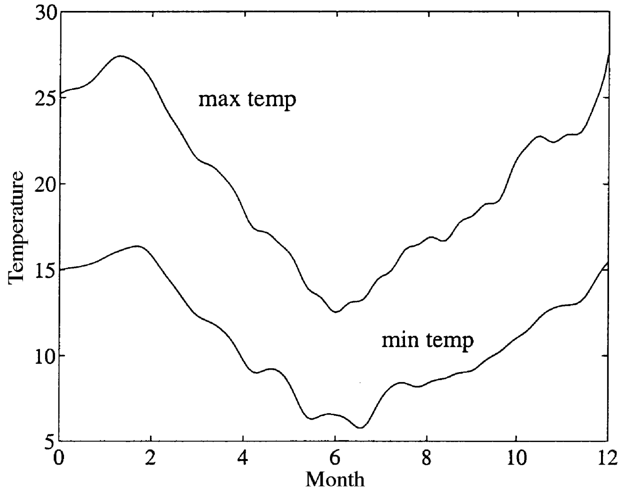

Figure 6.11 Prediction of maximum and minimum daily temperatures based on time of year. The model from Figure 6.10 implicitly learns a relationship between time of year and minimum/maximum temperature. This relationship is not directly invertible, but the temporal information used by extended Kalman smoothing correctly infers month given temperature as shown in Figure 6.10.

6.4 EXTENSIONS

6.4.1 Learning the Means and Widths of the RBFs

It is possible to relax the assumption that the Gaussian radial basis functions have fixed centers and widths, although this results in a somewhat more complicated and slower fitting algorithm. To derive learning rules for the RBF centers c_i and width matrices S_i, we need to consider how they play into the cost function (6.17) through the RBF kernel (6.14). We take derivatives with respect to the expectation of the cost function c, and exchange the order of the expectation and the derivative:

$$\left\langle \frac{\partial c}{\partial c_i} \right\rangle = \left\langle \frac{\partial c}{\partial \rho_i} \frac{\partial \rho_i}{\partial c_i} \right\rangle = 2\langle (\theta\Phi - z)^\top Q^{-1} h_i \rho_i(x) S_i^{-1}(x - c_i)\rangle. \tag{6.19}$$

Recalling that $\Phi = [\rho_1(x) \quad \rho_2(x) \quad \ldots \quad \rho_I(x) \quad x^\top \quad u^\top \quad 1]^\top$, it is clear that c_i figures nonlinearly in several places in this equation, and therefore it is not possible to solve for c_i in closed form. We can, however, use the above gradient to move the center c_i to decrease the cost, which corresponds to taking a *partial* M-step with respect to c_i. Equation (6.19) requires the computation of three third-order expectations in addition to the first- and second-order expectations needed to optimize θ and Q: $\langle \rho_i(x)\rho_k(x)x_l \rangle_j$, $\langle \rho_i(x)x_k x_l \rangle_j$, and $\langle \rho_i(x)z_k x_l \rangle_j$. Similarly, differentiating the cost with respect to S_i^{-1} gives

$$\left\langle \frac{\partial c}{\partial S_i^{-1}} \right\rangle = \left\langle \frac{\partial c}{\partial \rho_i} \frac{\partial \rho_i}{\partial S_i^{-1}} \right\rangle = \langle [(\theta\Phi - z)^\top Q^{-1} h_i] \rho_i(x)[S_i - (x - c_i)(x - c_i)^\top] \rangle. \tag{6.20}$$

We now need three *fourth*-order expectations as well: $\langle \rho_i(x)\rho_k(x)x_l x_m \rangle_j$, $\langle \rho_i(x)_k x_l x_m \rangle_j$, and $\langle \rho_i(x)z_k x_l x_m \rangle_j$.

These additional expectations increase both the storage and computation time of the algorithm – a cost that may not be compensated by the added advantage of moving of centers and widths by small gradient steps. One heuristic is to place centers and widths using unsupervised techniques such as the EM algorithm for Gaussian mixtures, which considers solely the input density and not the output nonlinearity. Alternatively, some of these higher-order expectations can be approximated using, for example, $\langle \rho_i(x) \rangle \approx \rho_i(\langle x \rangle)$.

6.4.2 On-line Learning

One of the major limitations of the algorithm that we have presented in this chapter is that it is a *batch* algorithm; that is, it assumes that we use the entire sequence of observations to estimate the model parameters. Fortunately, it is relatively straightforward to derive an on-line version of the algorithm, which updates parameters as it receives observations. This is achieved using the recursive least-squares (RLS) algorithm, which is in fact just a special case of the discrete Kalman filter (see, e.g., [8, 44]).

The key observation is that the cost minimized in the M-step of the algorithm (6.17) is a quadratic function of the parameters θ. RLS is simply a way of solving quadratic problems on-line. Using k to index time step, the resulting algorithm for scalar z is as follows:

$$\theta_k = \theta_{k-1} + (\langle z\Phi \rangle_k - \theta_{k-1} \langle \Phi\Phi^\top \rangle_k) P_k, \tag{6.21}$$

$$P_k = P_{k-1} - \frac{P_{k-1} \langle \Phi\Phi^\top \rangle_k P_{k-1}}{1 + \langle \Phi^\top P_{k-1} \Phi \rangle_k}, \tag{6.22}$$

$$Q_k = Q_{k-1} + \frac{1}{k} [\langle z^2 \rangle_k - \theta_k \langle \Phi z \rangle_k - Q_{k-1}]. \tag{6.23}$$

Let us ignore the expectations for now. Initializing $\theta_0 = 0$, $Q_0 = \mathbf{I}$, and P_0 very large, it is easy to show that, after a few iterations, the estimates of θ_k will rapidly converge to the exact values obtained by the least-squares solution. The estimate of Q will converge to the correct values plus a bias incurred by the fact that the early estimates of Q were based on residuals from θ_k rather than $\lim_{k\to\infty} \theta_k$. P_k is a recursive estimate of $(\sum_{j=1}^{k} \langle \Phi\Phi \rangle_j)^{-1}$, obtained by using the matrix inversion lemma.

There is an important way in which this on-line algorithm is an approximation to the batch EM algorithm we have described for nonlinear state-space models. The expectations $\langle \cdot \rangle_k$ in the online algorithm are computed by running a single step of the extended Kalman filter using the previous parameters θ_{k-1}. In the batch EM algorithm, the expectations are computed by running an extended Kalman *smoother* over the entire sequence using the current parameter estimate. Moreover, these expectations are used to re-estimate the parameters, the smoother is then re-run, the parameters are re-re-estimated, and so on, to perform the usual iterations of EM. In general, we can expect that, unless the time series is nonstationary, the parameter estimates obtained by the batch algorithm after convergence will model the data better than those obtained by the on-line algorithm.

Interestingly, the updates for the RLS on-line algorithm described here are very similar to the parameter updates used a dual extended Kalman filter approach to system identification [45] (see Chapter 5 and Section 6.5.5). This similarity is not coincidental, since, as mentioned, the Kalman filter can be derived as a generalization of the RLS algorithm. In fact, this similarity can be exploited in an elegant manner to derive an on-line algorithm for parameter estimation for nonstationary nonlinear dynamical systems.

6.4.3 Nonstationarity

To handle nonstationary time series, we assume that the parameters can drift according to a Gaussian random walk with covariance Σ_θ:

$$\theta_k = \theta_{k-1} + \epsilon_k, \qquad \text{where} \qquad \epsilon_k \sim \mathcal{N}(0, \Sigma_\theta).$$

As before, we have the following function relating the z variables to the parameters θ and nonlinear kernels Φ:

$$z_k = \theta_k \Phi_k + w_k, \qquad \text{where} \qquad w_k \sim \mathcal{N}(0, Q),$$

which we can view as the observation model for a "state variable" θ_k with time-varying "output matrix" Φ_k. Since both the dynamics and observation models are linear in θ and the noise is Gaussian, we can apply the following Kalman filter to recursively compute the distribution of drifting parameters θ:

$$\hat{\theta}_k = \hat{\theta}_{k-1} + \frac{(\langle z\Phi\rangle_k - \hat{\theta}_{k-1}\langle\Phi\Phi^\top\rangle_k)P_{k|k-1}}{Q_{k-1} + \langle\Phi^\top P_{k|k-1}\Phi\rangle_k}, \tag{6.24}$$

$$P_{k|k-1} = P_{k-1} + \Sigma_\theta, \tag{6.25}$$

$$P_k = P_{k|k-1} - \frac{P_{k|k-1}\langle\Phi\Phi^\top\rangle_k P_{k|k-1}}{Q_{k-1} + \langle\Phi^\top P_{k|k-1}\Phi\rangle_k}, \tag{6.26}$$

$$Q_k = Q_{k-1} + \lambda(\langle z^2\rangle_k - \hat{\theta}_k\langle\Phi z\rangle_k - Q_{k-1}). \tag{6.27}$$

There are two important things to note. First, these equations describe an ordinary Kalman filter, except that both the "output" z and "output matrix" Φ_k are jointly uncertain with a Gaussian distribution. Second, we have also assumed that the output noise covariance can drift by introducing a forgetting factor λ in its re-estimation equation. As before, the expectations are computed by running one step of the EKF over the hidden variables using θ_{k-1}.

While we derived this on-line algorithm starting from the batch EM algorithm, what we have ended up with appears almost identical to the dual extended Kalman filter (discussed in Chapter 5). Indeed, we have two Kalman filters – one extended and one ordinary – running in parallel, estimating the hidden states and parameters, respectively.

We can also view this on-line algorithm as an approximation to the Bayesian posterior over parameters and hidden variables. The true posterior would be some complicated distribution over the x, z, and θ parameters. Here, we have recursively approximated it with two independent Gaussians – one over (x, z) and one over θ. The approximated posterior for θ_k has mean $\hat{\theta}_k$ and covariance P_k.

6.4.4 Using Bayesian Methods for Model Selection and Complexity Control

Like any other maximum-likelihood procedure, the EM algorithm described in this chapter has the potential to overfit the data set – that is, to find spurious patterns in noise in the data, thereby generalizing poorly. In our implementation, we used some ridge regression, that is, a weight decay regularizer on the h_i parameters, which seemed to work well in practice but required some heuristics for setting regularization parameters. (Although, as mentioned previously, integrating over the hidden variables acts as a sort of modulated input noise, and so, in effect, performs ridge regression, which can eliminate the need for explicit regularization.)

A second closely related problem faced by maximum-likelihood methods is that there is no built-in procedure for doing model selection. That is, the value of the maximum of the likelihood is not a suitable way to choose between different model structures. For example, consider the problems of choosing the dimensionality of the state space x and choosing the number of basis functions I. Higher dimensions of x and more basis functions should always, in principle, result in higher maxima of the likelihood, which means that more complex models will always be preferred to simpler ones. But this, of course, leads to overfitting.

Bayesian methods provide a very general framework for simultaneously handling the overfitting and model selection problems in a consistent manner. The key idea of the Bayesian approach is to avoid maximization wherever possible. Instead, possible models, structures, parameters – in short, all settings of unknown quantities – should be weighted by their posterior probabilities, and predictions should be made according to this weighted posterior.

For our nonlinear dynamical system, we can, for example, treat the parameters θ as an unknown. Then the model's prediction of the output at time $k+1$ is

$$\begin{aligned}p(y_{k+1}|u_{1:k+1}, y_{1:k}) &= \int d\theta\, p(y_{k+1}|u_{k+1}, y_{1:k}, u_{1:k}, \theta)p(\theta|\, y_{1:k}, u_{1:k}) \\ &= \int d\theta\, p(\theta|\, y_{1:k}, u_{1:k}) \int dx_{k+1} p(y_{k+1}|u_{k+1}, x_{k+1}, \theta) \\ &\quad \times p(x_{k+1}|u_{1:k+1}, y_{1:k}, \theta),\end{aligned}$$

where the first integral on the last line is over the posterior distribution of the parameters and the second integral is over the posterior distribution of the hidden variables.

The posterior distribution over parameters can be obtained recursively from Bayes' rule:

$$p(\theta|\, y_{1:k}, u_{1:k}) = \frac{p(y_k|u_{1:k}, y_{1:k-1}, \theta)p(\theta|_{U1:k-1}, y_{1:k-1})}{p(y_k|u_{1:k}, y_{1:k-1})}.$$

The dual extended Kalman filter, the joint extended Kalman filter, and the nonstationary on-line algorithm from Section 6.4.3 are all coarse approximations of these Bayesian recursions.

The above equations are all implicitly conditioned on some choice of model structure s_m, that is, the dimension of x and the number of basis functions. Although the Bayesian modeling philosophy advocates averaging predictions of different model structures, if necessary it is also possible to use Bayes' rule to *choose* between model structures according to their probabilities:

$$P(s_m | y_{1:k}, u_{1:k}) = \frac{p(y_{1:k}|u_{1:k}, s_m)P(s_m)}{\sum_n p(y_{1:k}|u_{1:k}, s_n)P(s_n)}.$$

Tractable approximations to the required integrals can be obtained in several ways. We highlight three ideas, without going into much detail; an adequate solution to this problem for nonlinear dynamical systems requires further research. The first idea is the use of Markov-chain Monte Carlo (MCMC) techniques to sample over both parameters and hidden variables. Sampling can be an efficient way of computing high-dimensional integrals if the samples are concentrated in regions where parameters and states have high probability. MCMC methods such as Gibbs sampling have been used for linear dynamical systems [46, 47], while a promising method for nonlinear systems is particle filtering [18, 24], in which samples ("particles") can be used to represent the joint distribution over parameters and hidden states at each time step. The second idea is the use of so-called "automatic relevance determination" (ARD [48, 49]). This consists of using a zero-mean Gaussian prior on each parameter with tunable variances. Since these variances are parameters that control the prior distribution of the model parameters, they are referred to as *hyperparameters*. Optimizing these variance hyperparameters while integrating over the parameters results in "irrelevant" parameters being eliminated from the model. This occurs when the variance controlling a particular parameter goes to zero. ARD for RBF networks with a center on each data point has been used by Tipping [50] successfully for nonlinear regression, and given the name "relevance vector machine" in analogy to support vector machines. The third idea is the use of variational methods to lower-bound the model structure posterior probabilities. In exactly the same way that EM can be thought of as forming a lower bound on the likelihood using a distribution over the hidden variables, variational Bayesian methods lower-bound the evidence using a distribution over both hidden variables and parameters. Variational Bayesian methods have been used in [51] to infer the structure of linear dynamical systems, although the generalization to nonlinear systems of the kind described in this chapter is not straightforward.

Of course, in principle, the Bayesian approach would advocate averaging over all possible choices of c_i, S_i, I, Q, etc. It is easy to see how this can rapidly get very unwieldy.

6.5 DISCUSSION

6.5.1 Identifiability and Expressive Power

As we saw from the experiments described above, the algorithm that we have presented is capable of learning good density models for a variety of nonlinear time series. Specifying the class of nonlinear systems that our algorithm can model well defines its *expressive power*. A related question is: What is the ability of this model, in principle, to recover the actual parameters of specific nonlinear systems? This is the question of *model identifiability*. These two questions are intimately tied, since they both describe the mapping between actual nonlinear systems and model parameter settings.

There are three trivial degeneracies that make our model technically unidentifiable, but should not concern us. First, it is always possible to permute the dimensions in the state space and, by permuting the domain of the output mapping and dynamics in the corresponding fashion, obtain an exactly equivalent model. Second, the state variables can be rescaled or, in fact, transformed by any invertible linear mapping. This transformation can be absorbed by the output and dynamics functions, yielding a model with identical input–output behavior. Without loss of generality, we always set the covariance of the state evolution noise to be the identity matrix, which both sets the scale of the state space and disallows certain state transformations without reducing the expressive power of the model. Third, we take the observation noise to be uncorrelated with the state noise and both noises to be zero-mean, since, again without loss of generality, these can be absorbed into the f and g functions.[11]

There exist other forms of unidentifiability that are more difficult to overcome. For example, if both f and g are nonlinear, then (at least in the noise-free case), for any arbitrary invertible transformation of the state,

[11]Imagine that the joint noise covariance was nonzero: $\langle w_k v_k^\top \rangle = S$. Replacing A with $A' = A - SR^{-1}C$ gives a new noise process w' with covariance $Q' = Q - SR^{-1}S^\top$ that is uncorrelated with v, leaving the input–output behavior invariant. Similarly, any nonzero noise means can be absorbed into the b terms in the functions f and g.

there exist transformations of f and g that result in identical input–output behavior. In this case, it would he very hard to detect that the recovered model is indeed a faithful model of the actual system, since the estimated and actual states would appear to be unrelated.

Clearly, not all systems can be modeled by assuming that f is linear and g is nonlinear. Similarly, not all systems can be modeled by assuming that f is nonlinear and g is linear. For example, consider the case where the observations y_k and y_{k+n} are statistically independent, but each observation lies on a curved low-dimensional manifold in a high-dimensional space. Modeling this would require a nonlinear g as in nonlinear factor analysis, but an $f = 0$. Therefore, choosing either f or g to be linear restricts the expressive power of the model.

Unlike the state noise covariance Q, assuming that the observation noise covariance R is diagonal *does* restrict the expressive power of the model. This is easy to see for the case where the dimension of the state space is small and the dimension of the observation vector is large. A full covariance R can capture all correlations between observations at a single time step, while a diagonal R model cannot.

For nonlinear dynamical systems, the Gaussian-noise assumption is not as restrictive as it may initially appear. This is because the nonlinearity can be used to turn Gaussian noise into non-Gaussian noise [6].

Of course, we have restricted our expressive power by using an RBF network, especially one in which the means and centers of the RBFs are fixed. One could try to appeal to universal approximation theorems to make the claim that one could, in principle, model any nonlinear dynamical system. But this would be misleading in the light of the noise assumptions and the fact that only a finite and usually small number of RBFs are going to be used in practice.

6.5.2 Embedded Flows

There are two ways to think about the dynamical models we have investigated, shown in Figure 6.12. One is as a nonlinear Markov process (flow) x_k that has been embedded (or potentially projected) into a manifold y_k. From this perspective, the function f controls the evolution of the stochastic process, and the function g specifies the nonlinear embedding (or projection) operation.[12]

[12]To simplify presentation, we shall neglect driving inputs u_k in this section, although the arguments extend as well to systems with inputs.

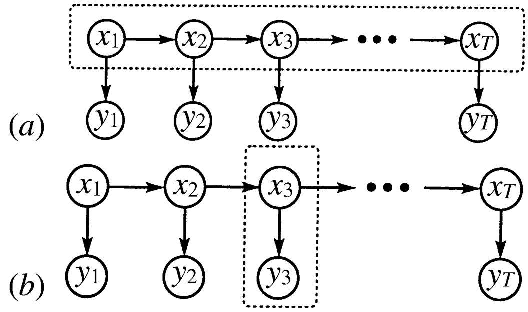

Figure 6.12 Two interpretations of the graphical model for stochastic (non)linear dynamical systems (see text). (*a*) A Markov process embedded in a manifold. (*b*) Nonlinear factor analysis through time.

Another way to think of the same model is as a nonlinear version of a latent-variable model such as factor analysis (but possibly with external inputs as well) in which the latent variables or factors evolve through time rather than being drawn independently for each observation. The nonlinear factor analysis model is represented by g and the time evolution of the latent variables by f.

If the state space is of lower dimension than the observation space and the observation noise is additive, then a useful geometrical intuition applies. In such cases, we have observed a flow inside an embedded manifold. The observation function g specifies the structure (shape) of the manifold, while the dynamics f specifies the flow within the manifold. Armed with this intuition, the learning problem looks as if it might be decoupled into two separate stages: first find the manifold by doing some sort of density modeling on the collection of observed outputs (ignoring their time order); second, find the flow (dynamics) by projecting the observations into the manifold and doing nonlinear regression from one time step to the next. This intuition is partly true, and indeed provides the basis for many of the practical and effective initialization schemes that we have tried. However, the crucial point as far as the design of learning algorithms is concerned is that the two learning problems interact in a way that makes the problem *easier.* Once we know something about the dynamics, this information gives some prior knowledge when trying to learn the manifold shape. For example, if the dynamics suggest that the next state will be near a certain point, we can use this information to do better than naive projection when we locate a noisy observation on the manifold. Conversely, knowing something about the manifold allows us to estimate the dynamics more effectively.

We discuss separately two special cases of flows in manifolds: systems with linear output functions but nonlinear dynamics, and systems with linear dynamics but nonlinear output function.

When the output function g is linear and the dynamics f is nonlinear (Fig. 6.13), the observed sequence forms a nonlinear flow in a linear subspace of the observation space. The manifold estimation is made easier, even with high levels of observation noise, by the fact that its shape is known to be a hyperplane. All that is required is to find its orientation and the character of the output noise. Time-invariant analysis of the observations by algorithms such as factor analysis is an excellent way to initialize estimates of the hyperplane and noises. However, during learning, we may have cause to tilt the hyperplane to make the dynamics fit better, or conversely cause to modify the dynamics to make the hyperplane model better.

This setting is actually more expressive than it might seem initially. Consider a nonlinear output function $g(x)$ that is "invertible" in the sense that it be written in the form $g(x) = C\tilde{g}(x)$ for invertible $\tilde{g}$ and non-square matrix C. Any such nonlinear output function can be made strictly linear if we transform to a new state variable $\tilde{x}$:

$$\tilde{x} = \tilde{g}(x) \Rightarrow \tilde{x}_{k+1} = \tilde{f}(\tilde{x}_k, w_k) = \tilde{g}(f(\tilde{g}^{-1}(\tilde{x})) + w_k), \qquad (6.28a)$$

$$y_k = C\tilde{x}_k + v_k = g(x_k) + v_k, \qquad (6.28b)$$

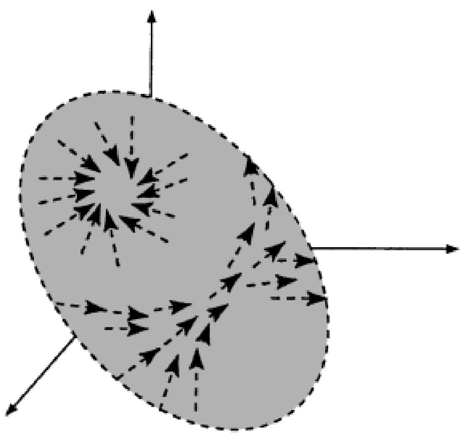

Figure 6.13 Linear and nonlinear dynamical systems represent flow fields embedded in manifolds. For systems with linear output functions, such as the one illustrated, the manifold is a hyperplane while the dynamics may be complex. For systems with nonlinear output functions the shape of the embedding manifold is also curved.

which gives an equivalent model but with a purely linear output process, and potentially nonadditive dynamics noise.

For nonlinear output functions g paired with linear dynamics f, the observation sequence forms a matrix (linear) flow in a nonlinear manifold:

$$x_{k+1} = Ax_k + w_k, \tag{6.29a}$$

$$y_k = g(x_k) + v_k. \tag{6.29b}$$

The manifold learning is harder now, because we must estimate a thin, curved subspace of the observation space in the presence of noise. However, once we have learned this manifold approximately, we project the observations into it and learn only linear dynamics. The win comes from the following fact: in the locations where the projected dynamics do not look linear, we know that we should bend the manifold to make the dynamics more linear. Thus, not only the shape of the outputs (ignoring time) but also the linearity of the dynamics give us clues to learning the manifold.

6.5.3 Stability

Stability is a key issue in the study of any dynamical system. Here we have to consider stability at two levels: the stability of the learning procedure, and the stability of the learned nonlinear dynamical system.

Since every step of the EM algorithm is guaranteed to increase the log-likelihood until convergence, it has a built-in Lyapunov function for stable learning. However, as we have pointed out, our use of extended Kalman smoothing in the E-step of the algorithm represents an approximation to the exact E-step, and therefore we have to forego any guarantees of stability of learning. While we rarely had problems with stability of learning, this is sure to be problem-specific, depending both on the quality of the EKS approximation and on how close the true system dynamics is to the boundary of stability. In contrast to the EKS approximations, certain variational approximations [29] transform the intractable Lyapunov function into a tractable one, and therefore preserve stability of learning. It is not clear how to apply these variational approximations to nonlinear dynamics, although this would clearly be an interesting area of research.

Stability of the learned nonlinear dynamical system can be analyzed by making use of some linear systems theory. We know that, for discrete-time linear dynamical systems, if all eigenvalues of the A matrix lie inside the unit circle, then the system is globally stable. The nonlinear dynamics of our RBF network f can be decomposed into two parts (cf. Eq. (6.13)): a linear component given by A, and a nonlinear component given by $\sum_i h_i \rho_i(x)$. Clearly, for the system to be globally stable, A has to satisfy the eigenvalue criterion for linear systems. Moreover, if the RBF coefficients for both f and g have bounded norm (i.e., $\max_i |h_i| < \bar{h}$) and the RBF is bounded, with $\min_i \det(S_i) > s_{\min} > 0$ and $\max_{ij} |c_i - c_j| < \bar{c}$, then the nonlinear system is stable in the following sense. The conditions on S_i and h mean that

$$\left| \sum_i h_i \rho_i(x) \right| < \frac{I\bar{h}}{(2\pi)^{d/2} \sqrt{s_{\min}}} \equiv \kappa.$$

Therefore, the noise-free nonlinear component of the dynamics alone will always maintain the state within a sphere of radius κ around $\bar{c}$. So, if the linear component is stable, then for any sequence of bounded inputs, the output sequence of the noise-free system will be bounded. Intuitively, although unstable behavior might occur in the region of RBF support, once x leaves this region it is drawn back in by A.

For the on-line EM learning algorithm, the hidden state dynamics and the parameter re-estimation dynamics will interact, and therefore a stability analysis would be quite challenging. However, since there is no stability guarantee for the batch EKS-EM algorithm, it seems very unlikely that a simple form of the on-line algorithm could be provably stable.

6.5.4 Takens' Theorem and Hidden States

It has long been known that for linear systems, there is an equivalence between so called state-space formulations involving hidden variables and direct vector autoregressive models of the time series. In 1980, Takens proved a remarkable theorem [52] that tells us that, for almost any deterministic nonlinear dynamical system with a d-dimensional state space, the state can be effectively reconstructed by observing $2d + 1$ time lags of any one of its outputs. In particular, Takens showed that such a lag vector will be a smooth embedding (diffeomorphism) of the true state, if one exists. This notion of finding an "embedding" for the state has been used to justify a nonlinear regression approach to learning nonlinear dynamical systems. That is, if you suspect that the system is nonlinear and that it has d state dimensions, then instead of building a state-space model, you can do away with representing states and just build an autoregressive (AR) model directly on the observations that nonlinearly relates previous outputs and the current output. (Chapter 4 discusses the case of chaotic dynamics.) This view begs the question: Do we need our models to have hidden states at all?

While no constructive realization for Takens' theorem exists in general, there are very strong results for linear systems. For purely linear systems, we can appeal to the Cayley–Hamilton theorem[13] to show that the hidden state can always be eliminated to obtain an equivalent vector autoregressive model by taking only d time lags of the output. Furthermore, there is a construction that allows this conversion to be performed explicitly.[14] Takens' theorem offers us a similar guarantee for elimination of hidden states in nonlinear dynamical systems, as long as we take $2d + 1$ output lags. (However, no similar recipe exists for explicitly converting to an autoregressive form). These results appear to make hidden states unnecessary.

The problem with this view is that it does not generalize well to many realistic high-dimensional and noisy scenarios. Consider the example mentioned in the introduction. While it is mathematically true that the pixels in the video frame of a balloon floating in the wind are a (highly nonlinear) function of the pixels in the previous video frames, it would be ludicrous from the modeling perspective to build an AR model of the video images. This would require a number of parameters of the order of the number of pixels squared. Furthermore, unlike the noise-free case of Takens' theorem, when the dynamics are noisy, the optimal prediction of the observation would have to depend on the entire history of past observations. Any truncation of this history throws away potentially valuable information about the unobserved state. The state-space formulation of nonlinear dynamical systems allows us to overcome both of these limitations of nonlinear autoregressive models. That is, it allows us to have compact representations of dynamics, and to integrate uncertain information over time. The price paid for this is that it requires inference over the hidden state.

[13]Any square matrix A of size n satisfies its own characteristic equation. Equivalently, any matrix power A^m for $m \geq n$ can be written as a linear combination of lower matrix powers $I, A, A^2, \ldots, A^{n-1}$.

[14]Start with the system $x_{k+1} = Ax_k + w_k$, $y_k = Cx_k + v_k$. Create a d-dimensionl lag vector $z_k = [y_k; y_{k+1}; \ldots; y_{k+d-1}]$ that holds the current and $d-1$ future outputs. Write $z_k = Gx_k + n_k$ for $G = [CI; CA; CA^2; \ldots; CA^{d-1}]$ and Gaussian noise n (although with nondiagonal covariance). The Cayley–Hamilton theorem assures us that G is full rank, and thus we need not take any more lags. Given the lag vector z_k, we can solve the system $z_k = Gx_k$ for x_k; write this solution as $G^+ z_k$. Using the original observation equation d times, to solve for $y_k, \ldots, y_{k+d-1}$ in terms of z_k, we can write an autoregression for z_k as $z_{k+1} = G^+ AGz_k + m_k$ for Gaussian noise m.

6.5.5 Should Parameters and Hidden States be Treated Differently?

The maximum-likelihood framework on which the EM algorithm is based makes a distinction between parameters and hidden variables: it attempts to *integrate* over hidden variables to *maximize* the likelihood as a function of parameters. This leads to the two-step approach, which computes sufficient statistics over the hidden variables in the E-step and optimizes parameters in the M-step. In contrast, a fully Bayesian approach to learning nonlinear dynamical state-space models would treat both hidden variables and parameters as unknown and attempt to compute or approximate the joint posterior distribution over them – in effect integrating over both.

It is important to compare these approaches to system identification with more traditional ones. We highlight two such approaches: *joint EKF* approaches and *dual EKF* approaches.

In joint EKF approaches [7, 8], an augmented hidden state space is constructed that comprises the original hidden state space and the parameters. Since parameters and hidden states interact, even for linear dynamical systems this approach results in nonlinear dynamics over the augmented hidden states. Initializing a Gaussian prior distribution both over parameters and over states, an extended Kalman filter is then used to recursively update the joint distribution over states and parameters based on the observations, $p(X, \theta|Y)$. This approach has the advantage that it can model uncertainties in the parameters and correlations between parameters and hidden variables. In fact, this approach treats parameters and state variables completely symmetrically, and can be thought of as iteratively implementing a Gaussian approximation to the recursive Bayes' rule computations. Nonstationarity can be easily built in by giving the parameters (e.g., random-walk) dynamics. Although it has some very appealing properties, this approach is known to suffer from instability problems, which is the reason why dual EKF approaches have been proposed.

In dual EKF approaches (see Chapter 5), two interacting but distinct extended Kalman filters run simultaneously. One computes a Gaussian approximation of the state posterior given a parameter estimate and the observations: $p(X|\hat{\theta}_{\text{old}}, Y)$, while the other computes a Gaussian approximation of the parameter posterior given the estimated states $p(\theta|\hat{X}_{\text{old}}, Y)$. The two EKFs interact by each feeding its estimate (i.e., the posterior means $\hat{X}$ and $\hat{\theta}$) into the other. One can think of the dual EKF as performing approximate coordinate ascent in $p(X, \theta|Y)$ by iteratively maximizing $p(X|\hat{\theta}_{\text{old}}, Y)$ and $p(\theta|\hat{X}_{\text{old}}, Y)$ under the assumption that each conditional is Gaussian. Since the only interaction between parameters and hidden variables occurs through their respective means, the procedure has the flavor of mean-field methods in physics and neural networks [53]. Like these methods, it is also likely to suffer from the overconfidence problem – namely, since the parameter estimate does not take into account the uncertainty in the states, the parameter covariance will be overly narrow, and likewise for the states.

For large systems, both joint and dual EKF methods suffer from the fact that the parameter covariance matrix is quadratic in the number of parameters. This problem is more pronounced for the joint EKF, since it considers the concatenated state space. Furthermore, both joint and dual EKF methods rely on Gaussian approximations to parameter distributions. This can sometimes be problematic – for example, consider retaining positive-definiteness of a noise covariance matrix under the assumption that its parameters are Gaussian-distributed.

6.6 CONCLUSIONS

This chapter has brought together two classic algorithms – one from statistics and another from systems engineering – to address the learning of stochastic nonlinear dynamical systems. We have shown that by pairing the extended Kalman smoothing algorithm for approximate state estimation in the E-step with a radial basis function learning model that permits exact analytic solution of the M-step, the EM algorithm is capable of learning a nonlinear dynamical model from data. As a side-effect we have derived an algorithm for training a radial basis function network to fit data in the form of a mixture of Gaussians. We have also derived an on-line version of the algorithm and a version for dealing with nonstationary time series.

We have demonstrated the algorithm on a series of synthetic and realistic nonlinear dynamical systems, and have shown that it is able to learn accurate models from only observations of inputs and outputs. Initialization of model parameters and placement of the radial basis kernels are important to the practical success of the algorithm. We have discussed techniques for making these choices, and have provided gradient rules for adapting the centers and widths of the basis functions.

The main strength of our algorithm is that by making a specific choice of nonlinear estimator (Gaussian radial basis networks), we are able to exactly account for the uncertain state estimates generated during inference. Furthermore, the parameter-update procedures still only require the solution of systems of linear equations. However, we rely on the standard, but potentially inaccurate, extended Kalman smoother for approximate inference. For certain problems where local linearization is an extremely poor approximation, greater accuracy may be achieved using other approximate inference techniques such as the unscented filter (see Chapter 7). Another area worthy of further investigation is how to initialize the parameters more effectively when the data lie on a nonlinear manifold; in these cases, factor analysis provides an inadequate static model.

The belief network literature has recently been dominated by two methods for approximate inference: Markov-chain Monte Carlo [54] and variational approximations [29]. To the best of our knowledge, [36] and [45] were the first instances where extended Kalman smoothing was used to perform approximate inference in the E-step of EM. While EKS does not have the theoretical guarantees of variational methods (which are also approximate, but monotonically optimize a computable objective function during learning), its simplicity has gained it wide acceptance in the estimation and control literatures as a method for doing inference in nonlinear dynamical systems. Our practical success in modeling a variety of nonlinear time series suggests that the combination of extended Kalman algorithms and the EM algorithm can provide powerful tools for learning nonlinear dynamical systems.

ACKNOWLEDGMENTS

The authors acknowledge support from the Gatsby Charitable Fund, and thank Simon Haykin for helpful comments that helped to clarify the presentation. S.R. is supported in part by a National Science Foundation PDF Fellowship and a grant from the Natural Sciences and Engineering Research Council of Canada.

APPENDIX: EXPECTATIONS REQUIRED TO FIT THE RBFs

The expectations that we need to compute for Eq. (6.78) are $\langle x \rangle_j$, $\langle z \rangle_j$, $\langle xx^\top \rangle_j$, $\langle zz^\top \rangle_j$, $\langle xz^\top \rangle_j$, $\langle \rho_i(x) \rangle_j$, $\langle x\rho_i(x) \rangle_j$, $\langle zp_i(x) \rangle_j$, and $\langle \rho_i(x)\rho_l(x) \rangle_j$. Starting with some of the easier ones that do not depend on the RBF kernel ρ, we have

$$\langle x \rangle_j = \mu_j^x, \qquad \langle z \rangle_j = \mu_j^z,$$
$$\langle xx^\top \rangle_j = \mu_j^x \mu_j^{x,\top} + C_j^{xx}, \qquad \langle zz^\top \rangle_j = \mu_j^z \mu_n^{z,\top} + C_j^{zz},$$
$$\langle xz^\top \rangle_j = \mu_j^x \mu_j^{z,\top} + C_j^{xz}.$$

Observe that when we multiply the Gaussian RBF kernel $\rho_i(x)$ (Eq. (6.14)) and $\mathcal{N}_j$, we get a Gaussian density over (x, z) with mean and covariance

$$\mu_{ij} = C_{ij}\left(C_j^{-1}\mu_j + \begin{bmatrix} S_i^{-1}c_i \\ 0 \end{bmatrix}\right), \qquad C_{ij} = \left(C_j^{-1} + \begin{bmatrix} S_i^{-1} & 0 \\ 0 & 0 \end{bmatrix}\right)^{-1},$$

and an extra constant (due to lack of normalization),

$$\beta_{ij} = (2\pi)^{-d_x/2}|S_i|^{-1/2}|C_j|^{-1/2}|C_{ij}|^{1/2}\exp(-\tfrac{1}{2}\delta_{ij}),$$

where

$$\delta_{ij} = c_i^\top S_i^{-1}c_i + \mu_j^\top C_j^{-1}\mu_j - \mu_{ij}^\top C_{ij}^{-1}\mu_{ij}.$$

Using β_{ij} and μ_{ij}, we can evaluate the other expectations:

$$\langle\rho_i(x)\rangle_j = \beta_{ij}, \qquad \langle x\rho_i(x)\rangle_j = \beta_{ij}\mu_{ij}^x, \qquad \langle z\rho_i(x)\rangle_j = \beta_{ij}\mu_{ij}^z.$$

Finally,

$$\langle\rho_i(x)\rho_l(x)\rangle_j = (2\pi)^{-d_x}|C_j|^{-1/2}|S_i|^{-1/2}|S_l|^{-1/2}|C_{ilj}|^{1/2}\exp(\tfrac{1}{2}\gamma_{ilj}),$$

where

$$C_{ilj} = \left(C_j^{-1} + \begin{bmatrix} S_i^{-1} + S_l^{-1} & 0 \\ 0 & 0 \end{bmatrix}\right)^{-1},$$

$$\mu_{ilj} = C_{ilj}\left(C_j^{-1}\mu_j + \begin{bmatrix} S_i^{-1}c_i + S_l^{-1}c_l \\ 0 \end{bmatrix}\right),$$

$$\gamma_{ilj} = c_i^\top S_i^{-1}c_i + c_l^\top S_l^{-1}c_l + \mu_j^\top C_j^{-1}\mu_j - \mu_{ilj}^\top C_{ilj}^{-1}\mu_{ilj}.$$

REFERENCES

[1] R.E. Kalman and R.S. Bucy, "New results in linear filtering and prediction," *Transactions of the ASME Ser. D, Journal of Basic Engineering*, **83**, 95–108 (1961).

[2] L.E. Baum and J.A. Eagon, "An equality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology," *Bulletin of the American Mathematical Society*, **73**, 360–363 (1967).

[3] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Ser. B*, **39**, 1–38 (1977).

[4] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann, 1988.

[5] S.L. Lauritzen and D.J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems," *Journal of the Royal Statistical Society, Ser. B*, **50**, 157–224 (1988).

[6] S. Roweis and Z. Ghahramani, "A unifying review of linear Gaussian models," *Neural Computation*, **11**, 305–345 (1999).

[7] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*. Cambridge, MA: MIT Press, 1983.

[8] G.C. Goodwin and K.S. Sin, *Adaptive Filtering, Prediction and Control*. Englewood Cliffs, NJ: Prentice-Hall, 1984.

[9] J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, **1**, 281–294 (1989).

[10] D.S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, **2**, 321–355 (1988).

[11] R.E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, **82**, 35–45 (1960).

[12] H.E. Rauch, "Solutions to the linear smoothing problem," *IEEE Transactions on Automatic Control*, **8**, 371–372 (1963).

[13] R.E. Kopp and R.J. Orford, "Linear regression applied to system identification and adaptive control systems," *AIEE Journal*, **1**, 2300–2306 (1963).

[14] H. Cox, "On the estimation of state variables and parameters for noisy dynamic systems." *IEEE Transactions on Automatic Control*, **9**, 5–12 (1964).

[15] S. Singhal and L. Wu, "Training multiplayer perceptrons with the extended Kalman algorithm," in *Advances in Neural Information Processing Systems*, Vol. 1, San Mateo, CA: Morgan Kaufmann, 1989, 133–140.

[16] V. Kadirkamanathan and M. Niranjan, "A functional estimation approach to sequential learning with neural networks," *Neural Computation*, **5**, 954–975 (1993).

[17] I.T. Nabney, A. McLachlan, and D. Lowe, "Practical methods of tracking of nonstationary time series applied to real-world data," in S. K. Rogers and D. W. Ruck, Eds. *AeroSense '96: Applications and Science of Artificial Neural Networks II*. SPIE Proceedings, 1996, pp. 152–163.

[18] J.E. Handschin and D.Q. Mayne, "Monte Carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering," *International Journal of Control*, **9**, 547–559 (1969).

[19] N.J. Gordon, D.J. Salmond, and A.F.M. Smith, "A novel approach to nonlinear/non-Gaussian Bayesian state space estimation," *IEE Proceedings F: Radar and Signal Processing*, **140**, 107–113 (1993).

[20] G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models," *Journal of Computer Graphics and Graphical Statistics*, **5**, 1–25 (1996).

[21] M. Israd and A. Blake, "CONDENSATION – conditional density propagation for visual tracking," *International Journal of Computer Vision*, **29**, 5–28 (1998).

[22] M. West, "Approximating posterior distributions by mixtures," *Journal of the Royal Statistical Society, Ser, B*, **54**, 553–568 (1993).

[23] M. West, "Mixture models, monte carlo, Bayesian updating and dynamic models," *Computing Science and Statistics*, **24**, 325–333 (1993).

[24] A. Doucet, J.F.G. de Freitas, and N.J. Gordon, *Sequential Monte Carlo Methods in Practice*. New York: Springer-Verlag, 2000.

[25] S.J. Julier, J.K. Uhlmann, and H.F. Durrant-Whyte, "A new approach for filtering nonlinear systems," in *Proceedings of the 1995 American Control Conference, Seattle, 1995*, pp. 1628–1632.

[26] S.J. Julier and J.K. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in *Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Orlando, FL, 1997*.

[27] E.A. Wan, R. van der Merwe, and A.T. Nelson, "Dual estimation and the unscented transformation," in *Advances in Neural Information Processing Systems*, Vol. 12, Cambridge, MA: MIT Press, 1999.

[28] Z. Ghahramani and G.E. Hinton, "Variational learning for switching state-space models," *Neural Computation*, **12**, 831–864 (2000).

[29] M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, and L.K. Saul, "An introduction to variational methods in graphical models," *Machine Learning*, **37**, 183–233 (1999).

[30] L.E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains," *Annals of Mathematical Statistics*, **37**, 1554–1563 (1966).

[31] R.M. Neal and G.E. Hinton, "A view of the EM algorithm that justifies incremental sparse and other variants," in M.I. Jordan, ed., *Learning in Graphical Models*. Dordrecht: Kluwer Academic, 1998, pp. 355–368.

[32] I. Csiszár and G. Tusnády, "Information geometry and alternating minimization procedures," *Statistics & Decisions*, Supplement Issue 1, pp. 205–237, 1984.

[33] C.F. Chen, "The EM approach to the multiple indicators and multiple causes model via the estimation of the latent variables." *Journal of the American Statistical Association*, **76**, 704–708 (1981).

[34] R.H. Shumway and D.S. Stoffer, "An approach to time series smoothing and forecasting using the EM algorithm," *Journal of Time Series Analysis*, **3**, 253–264 (1982).

[35] Z. Ghahramani and G. Hinton, "Parameter estimation for linear dynamical systems," Technical Report CRG-TR-96-2, Department of Computer Science, University of Toronto, February 1996.

[36] Z. Ghahramani and S. Roweis, "Learning nonlinear dynamical systems using an EM algorithm," in *Advances in Neural Information Processing Systems*, Vol. 11. Cambridge, MA: MIT Press, 1999, pp. 431–437.

[37] J.F.G. de Freitas, M. Niranjan, and A.H. Gee, "Nonlinear state space estimation with neural networks and the EM algorithm," Technical Report, Cambridge University Engineering Department, 1999.

[38] T. Briegel and V. Tresp, "Fisher scoring and a mixture of modes approach for approximate inference and learning in nonlinear state space models," in *Advances in Neural Information Processing Systems*, Vol. 11. Cambridge, MA, MIT Press, 1999.

[39] Z. Ghahramani and G. Hinton "Switching state-space models," Technical Report CRG-TR-96-3, Department of Computer Science, University of Toronto, July 1996.

[40] K. Murphy, "Switching Kalman filters," Technical Report, Department of Computer Science, University of California, Berkeley, August, 1998.

[41] G.E. Hinton, P. Dayan, and M. Revow, "Modeling the manifolds of Images of handwritten digits," *IEEE Transactions on Neural Networks*, **8**, 65–74 (1997).

[42] Z. Ghahramani and G. Hinton, "The EM algorithm for mixtures of factor analyzers," Technical Report CRG-TR-96-1, Department of Computer Science, University of Toronto, May 1996 (revised February 1997).

[43] W.S. Torgerson, "Multidimensional scaling I. Theory and method," *Psychometrika*, **17**, 401–419 (1952).

[44] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Upper Saddle River, NJ: Prentice-Hall, 1996.

[45] E.A. Wan and A.T. Nelson, "Dual Kalman filtering methods for nonlinear prediction," in *Advances in Neural Information Processing Systems*, Vol. 9. Cambridge, MA: MIT Press, 1997.

[46] C.K. Carter and R. Kohn, "On Gibbs sampling for state space models," *Biometrika*, **81**, 541–553 (1994).

[47] S. Früwirth-Schnatter, "Bayesian model discrimination and Bayes factors for linear Gaussian state space models." *Journal of the Royal Statistical Society, Ser. B*, **57**, 237–246 (1995).

[48] D.J.C. MacKay, "Bayesian non-linear modelling for the prediction competition," *ASHRAE Transcations*, **100**, 1053–1062 (1994).

[49] R.M. Neal, "Assessing relevance determination methods using DELVE," in C.M. Bishop, Ed. *Neural Networks and Machine Learning*. New York: Springer-Verlag, 1998, pp. 97–129.

[50] M.E. Tipping, "The relevance vector machine," in *Advances in Neural Information Processiing Systems*, Vol. 12. Cambridge, MA: MIT Press, 2000, pp. 652–658.

[51] Z. Ghahramani and M.J. Beal, "Propagation algorithms for variational Bayesian learning." in *Advances in Neural Information Processing Systems*, Vol. 13. Cambridge, MA: MIT Press, 2001.

[52] F. Takens, "Detecting strange attractors in turbulence," in D.A. Rand and L.-S. Young, Eds., *Dynamical Systems and Turbulence, Warwick 1980*, Lecture Notes in Mathematics, Vol. 898. Berlin: Springer-Verlag, 1981, pp. 365–381.

[53] J. Hertz, A. Krogh, and R.G. Palmer, *Introduction to the Theory of Neural Computation*. Redwood City, CA: Addison-Wesley, 1991.

[54] R.M. Neal, "Probablistic inference using Markov chain Monte Carlo methods," Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, 1993.