Mixtures of Predictive Linear Gaussian Models for Nonlinear Stochastic Dynamical Systems

David Wingate and **Satinder Singh**
Computer Science and Engineering Department
University of Michigan, Ann Arbor, MI 48109
{wingated,baveja}@umich.edu

Abstract

The Predictive Linear Gaussian model (or PLG) improves upon traditional linear dynamical system models by using a *predictive* representation of state, which makes consistent parameter estimation possible without any loss of modeling power and while using fewer parameters. This work extends the PLG to model nonlinear dynamical systems through the use of a kernelized, nonlinear mixture technique. The resulting generative model has been named the "MPLG," for "Mixture of PLGs." We also develop a novel technique to perform inference in the model, which consists of a hybrid of sigma-point approximations and analytical statistics. We show that the technique leads to fast and accurate approximations, and that it is general enough to be applied in other contexts. We empirically explore the MPLG and demonstrate its viability on several real-world and synthetic tasks.

Introduction

Model building is an important part of AI. Many agent-environment interactions can be modeled as dynamical systems, including such things as the dynamics of a biped walking or the trajectory of a ball bouncing. Traditional dynamical systems are modeled using the concept of state, which represents a sufficient statistic for history. Recently, however, a *predictive* perspective has been successfully taken. In a predictive model, the usual notion of state is represented by a set of (possibly action-conditional) predictions about the future. This is permissible because the most general definition of "state" is *any* sufficient statistic for history—in the predictive case, a finite set of predictions about the future summarize an infinite past.

These predictive representations have enjoyed considerable success. In the case of discrete observations, for example, PSRs (Littman, Sutton, & Singh 2001) have been shown to be just as flexible, expressive and compact as POMDPs. In the case of continuous observations and linear dynamics, the Predictive Linear Gaussian model (or PLG) is just as flexible, expressive and more compact than the celebrated Kalman filter, and strictly more powerful than ARMA (autoregressive) models (Rudary, Singh, & Wingate 2005).

However, the PLG is limited to *linear* dynamical systems. The primary contribution of this work is to extend the PLG to *nonlinear*, stochastic dynamical systems with a mixture approach. We contribute a fully generative model, which we have named the "Mixture of PLGs" (or MPLG). We compare this model to another nonlinear extension of the PLG, called the "Kernel PLG," and discuss why the MPLG is expected to generalize better and why it is expected to have superior representational power. We additionally compare to two nonlinear autoregressive models, and show that n predictions about the future are a very different thing than n memories about the past; these n predictions effectively give our model the infinite memory associated with state-space models.

Another contribution is a new inference technique we call "hybrid particle-analytical inference." Because our model is defined in terms of random variables, certain statistics of these variables must be computed to update state. Because the needed functions are nonlinear, exact analytical inference is generally impossible. This motivates some sort of approximation technique, so we have chosen *sigma-point approximations* (SPAs; also called "unscented transformations"), which have been introduced in the context of nonlinear Kalman filtering (Julier & Uhlmann 1996). This work broadens the utility and efficiency of SPAs by contributing (and leveraging) the insight that they are effectively built around the smoothing property of conditional expectations. Standard SPAs sample from all of the random variables in a model simultaneously, but sampling only a subset of variables can lead to significant computational advantage: part of our model is approximated with sigma-points, but *given* those sigma-points, exact analytical inference is possible on the rest of the model. This hybrid technique is a general method, and could be applied in other contexts.

After introducing the model and our hybrid technique, we show how the model's mixture perspective leads to natural parameter estimators, which are kernel-weighted versions of the original PLG estimators. We end by empirically comparing the proposed model to two nonlinear alternatives, and conclude that our proposed model exhibits an advantage over all of them, and in particular, over n-th order autoregressive models. This implies, as we will argue throughout the paper, that predictions about the future can constitute state, and give our model an effectively infinite memory.

Background: Predictive Gaussian Systems

Here, we briefly review the important concepts and definitions associated with linear dynamical systems and Predictive Gaussian Systems. Since the PLG, MPLG and Kernel PLG (another PLG-based model defined in [Wingate and Singh, 2006], which is reviewed later) all represent and update state in the same way, we discuss them collectively first.

Linear Dynamical Systems

A discrete time, linear dynamical system (LDS) is defined by a state update equation $x_{t+1} = Ax_t + \eta$, where $x_t \in \mathbb{R}^n$ is the state at time t, $A \in \mathbb{R}^{n\times n}$ is a transition matrix and $\eta \sim \mathcal{N}(0, Q)$ is mean-zero Gaussian noise (where $Q \in \mathbb{R}^{n\times n}$ is a covariance matrix). Often, we are not able to observe the state directly. Many LDSs define a companion observation process, in which observations are linear functions of the true state: $y_t = Hx_t + \mathcal{N}(0, R)$, where $H \in \mathbb{R}^{m\times n}$ and $R \in \mathbb{R}^{m\times m}$. There are generally no restrictions on H; in particular, it may collapse an n-dimensional state into a lower-dimensional (or even scalar) observation.

Predictive Gaussian Systems

We call either the PLG, the MPLG, or the Kernel PLG (KPLG) a *Predictive Gaussian System.* In Predictive Gaussian systems, we never refer to an unobservable or latent state x_t. Instead, we capture state as statistics about a random variable Z_t, which is defined as a vector of random variables predicting *future* observations. We associate each future observation at time $t + i$ with a random variable Y_{t+i} (all observations are scalars), and collect the next n of them into the vector $Z_t = [Y_{t+1} \cdots Y_{t+n}]^T$, as illustrated in Figure 1. These n variables are jointly Gaussian, with mean μ_t and covariance Σ_t. It is these two statistics that are used as the state of the system.

The system dynamics are defined by a special equation:

$$Y_{t+n+1} = f(Z_t, \eta_{t+n+1}) \tag{1}$$

where $\eta_{t+n+1} \in \mathbb{R}$ is a special noise term. The importance of modeling Y_{t+n+1} as a function of Z_t will be explained in the next section. In the PLG, Y_{t+n+1} is a *linear* function of Z_t, which allows it to model linear dynamical systems. In the MPLG and the KPLG, however, Y_{t+n+1} is a *nonlinear* function of Z_t, which allows them to model nonlinear dynamics.

The noise term is mean-zero with a fixed variance: $\eta_{t+n+1} \sim \mathcal{N}(0, \sigma^2_\eta)$, but is allowed to covary with the next n observations in a way that is independent of history: $\mathrm{Cov}[Z_t, \eta_{t+n+1}] = C_\eta$. The representational power of Predictive Gaussian Systems comes from this noise term: the fact that it covaries with future observations gives it the infinite memory of the LDS—an observation can have an effect far in the future through the chain of influence created by the correlation in the noise terms. Later, we will see that the differences in this noise term are one of the primary differences between the MPLG and the KPLG.

Updating State: Extend and Condition

We will now discuss the general strategy of Predictive Gaussian Systems for updating state and modeling dynamical systems, as well as why Y_{t+n+1} is modeled as a function of Z_t. Here, we restrict ourselves to scalar observations, but emphasize that this does not restrict the dimensionality of the underlying state space.

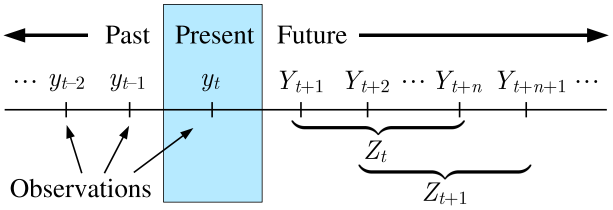

Figure 1: Timeline illustrating the random variables we use.

Modeling the system dynamics requires determining how to update the state of the system. The problem can be stated thus: given a state at time t, how can we incorporate an observation $Y_{t+1} = y_{t+1}$ to compute our state at time $t + 1$? The strategy is to *extend and condition*, as follows.

We begin with state extension. We assume that we have the state at time t, represented by μ_t and Σ_t. These statistics describe $Z_t \sim \mathcal{N}(\mu_t, \Sigma_t)$, which is an $n-$dimensional Gaussian describing the next n observations. We will extend this variable to include the variable Y_{t+n+1} (ensuring that it is still jointly Gaussian), creating a temporary $(n + 1)$-dimensional Gaussian. In order to extend Z_t to include the variable Y_{t+n+1}, we must compute three terms, which are $E_t = \mathrm{E}[Y_{t+n+1}]$, $C_t = \mathrm{Cov}[Y_{t+n+1}, Z_t]$ and $V_t = \mathrm{Var}[Y_{t+n+1}]$:

$$\left(\begin{array}{c} Z_t \\ Y_{t+n+1} \end{array}\right) \sim \mathcal{N}\left[\left(\begin{array}{c} \mu_t \\ E_t \end{array}\right), \left(\begin{array}{cc} \Sigma_t & C_t \\ C_t^T & V_t \end{array}\right)\right].$$

We will then condition on the observation y_{t+1}, which will result in another $n-$dimensional Gaussian RV describing $[Y_{t+2} \cdots Y_{t+n+1}]^T = Z_{t+1}$ (conditioning is done with standard techniques on multivariate Gaussians, for which it is well-known that the resulting RV is Gaussian). This results in $\mathrm{E}[Y_{t+2} \cdots Y_{t+n+1}] = \mathrm{E}[Z_{t+1}] = \mu_{t+1}$, along with $\mathrm{Cov}[Z_{t+1}] = \Sigma_{t+1}$, which are precisely the statistics representing our new state. Figure 1 illustrates Z_t and Z_{t+1}.

Rolling together the construction of the temporary Gaussian and the conditioning yields the complete state update:

$$\mu_{t+1} = \mu^-_{t+1} + K_t(y_{t+1} - e_1^T \mu_t) \tag{2}$$

$$\Sigma_{t+1} = (I - K_t e_1^T)\Sigma^-_{t+1} \tag{3}$$

where $\Sigma^-_{t+1} = I^-\Sigma_t I^{-T} + I^- C_t + C_t^T I^{-T} + e_n e_n^T V_t$, $K_t = \Sigma^-_{t+1} e_1 (e_1^T \Sigma_t e_1)^{-1}$, $\mu^-_{t+1} = I^-\mu_t + e_n E_t$,

$$I^- = \left(\begin{array}{c|c} \mathbf{0} & I_{n-1} \\ \hline \multicolumn{2}{c}{\mathbf{0}} \end{array}\right),$$

and e_i is the i-th column of the identity matrix. Note that Eqs. (2) and (3) have the same form as the Kalman filter.

Computing E_t, C_t and V_t in closed form for an arbitrary function $f()$ is impossible, which motivates the use of an approximation. The section on sigma-point approximations therefore presents a method which can be used for any $f()$.

Linear Dynamics: The PLG

In the PLG, Y_{t+n+1} is defined to be a linear function of Z_t:

$$Y_{t+n+1} = \langle g, Z_t \rangle + b + \eta_{t+n+1}. \tag{4}$$

where $\langle \cdot \rangle$ denotes inner product and $g \in \mathbb{R}^n$ is the linear trend, and where we have augmented the original PLG with a scalar bias term b. Rudary et al. (2005) showed how the state of this system can be recursively updated in closed form, and that it is equivalent in modeling power to the Kalman filter.

The MPLG: A Mixture of PLGs

We now present the MPLG, or Mixture of PLGs model. Consider the following scenario. Suppose that at time t, we have J PLGs, each with different parameters, and each specifying a different distribution over Y_{t+n+1}. We wish to combine them together to form a composite estimate of Y_{t+n+1}; the natural way to do this would be with a weighted sum:

$$Y_{t+n+1} = \sum_{j=1}^{J} w(Z_t)_j \left(\langle g^j, Z_t \rangle + b^j + \eta^j_{t+n+1} \right), \tag{5}$$

where the $w(Z_t)_j$'s are the mixing weights. It is important to note that these weights should be a nonlinear function of Z_t, because if they were linear they could simply be absorbed into the g^j's, resulting in a linear model. They should also sum to one, to maintain a well-defined mixture semantic. Here, $\mathrm{Var}[\eta^j_{t+n+1}] = \sigma^{2j}_{\eta}$ and $\mathrm{Cov}[\eta^j_{t+n+1}, Z_t] = C^j_{\eta}$.

A Distribution Over PLGs

Eq. (5) is the general form of the MPLG. We will now make specific choices about the function $w()$. We start by creating a new random variable $\mathbb{Y}_t$ that describes a distribution over possible Y_{t+n+1}'s; each Y_{t+n+1} is itself a Gaussian random variable describing distributions over actual observations y_{t+n+1}. We then specify a joint distribution over $\mathbb{Y}_t$ and Z_t, and use conditional expectation and a density over possible models to arrive at the final mixture of PLGs.

Suppose we use a Parzen kernel estimator to represent the joint density of $\mathbb{Y}_t$ and Z_t; suppose further that we use Gaussian kernels. Such an estimator would take the form:

$$p(\mathbb{Y}_t, Z_t) = \frac{1}{J} \sum_{j=1}^{J} \frac{1}{c_j} K(\mathbb{Y}_t, \xi_{\mathbb{Y}_t}; \phi_j) \frac{1}{c_j} K(Z_t, \xi_j; \phi_j) \tag{6}$$

where $1/c_j$ is a standard Gaussian normalizer and $K(x, y; \phi)$ is a Gaussian function with covariance matrix ϕ. The $\xi_j \in \mathbb{R}^n$ are points that could come from a number of sources: they may come from training data, be derived analytically, or be randomly generated. The $\xi_{\mathbb{Y}_t}$ variables will disappear in the following derivation.

We can use this estimator to derive the MPLG as shown below (derivation adapted from Bishop, 1995; pg. 178). In the fourth line, we will use the Parzen estimator of the joint probabilities (several terms cancel); note that this resembles the well-known Nadaraya-Watson estimator. In the fifth line, we replace each Y^j_{t+n+1} with a PLG that generates it, and in the final line, we summarize the kernel renormalization into a vector of weights $w(Z_t)$. As required, these weights are a nonlinear function of Z_t and sum to one:

$$\begin{aligned}
Y_{t+n+1} &= \mathrm{E}[\mathbb{Y}_t | Z_t] \\
&= \int \mathbb{Y}_t p(\mathbb{Y}_t | Z_t) d\mathbb{Y}_t \\
&= \frac{\int \mathbb{Y}_t p(\mathbb{Y}_t, Z_t) d\mathbb{Y}_t}{\int p(\mathbb{Y}_t, Z_t) d\mathbb{Y}_t} \\
&= \frac{\sum_{j=1}^{J} K(\xi_j, Z_t; \phi_j) Y^j_{t+n+1}}{\sum_{j=1}^{J} K(\xi_j, Z_t; \phi_j)} \\
&= \sum_{j=1}^{J} \frac{K(\xi_j, Z_t; \phi_j)}{\sum_{k=1}^{J} K(\xi_k, Z_t; \phi_k)} \left(\langle g^j, Z_t \rangle + b^j + \eta^j_{t+n+1} \right) \\
&= \sum_{j=1}^{J} w(Z_t)_j \left(\langle g^j, Z_t \rangle + b^j + \eta^j_{t+n+1} \right)
\end{aligned}$$

This leads us to the final MPLG model:

$$Y_{t+n+1} = \sum_{j=1}^{J} \frac{K(\xi_j, Z_t; \phi_j)}{\sum_{k=1}^{J} K(\xi_k, Z_t; \phi_k)} \left(\langle g^j, Z_t \rangle + b^j + \eta^j_{t+n+1} \right). \tag{7}$$

We can think of this as J PLGs, each centered at some ξ_j, and with the kernels acting as a distance metric between Z_t and ξ_j. How might such a model behave? Figure 2 builds some intuition: near the Gaussian centers, the individual PLGs are nonlinearly mixed. Further away from the centers, a single PLG becomes responsible for the space, resulting in a linear function. This contrasts sharply with a mixture of un-renormalized Gaussians: as we go further away from *their* centers, the function defining Y_{t+n+1} would go to zero.

Comparison to Related Models

We now compare the MPLG to two other models. The first is the Kernel PLG, which is derived by rewriting the PLG in dual form. The second is a kernel autoregressive model.

The Kernel PLG, or KPLG, defines the state extension as

$$Y_{t+n+1} = \textstyle\sum_{j=1}^{J} \alpha_j K(\xi_j, Z_t) + \eta_{t+n+1}, \tag{8}$$

where $K()$ is our kernel. This is the most obvious way to kernelize the original PLG algorithm, because the linear trend g has been rewritten in the dual form (that is, as a weighted combination of data points ξ_j).

There are two reasons the KPLG is insufficient to replace the PLG. First, the MPLG is expected to generalize the dynamics better outside of the training region, especially when using Gaussian kernels. Figure 2 illustrates this: far away from the ξ_j's, the MPLG generalizes linearly, but the KPLG model in Eq. (8) will return something close to zero.

Second, the term η_{t+n+1} in the KPLG has the same properties as in the PLG, and in particular $\mathrm{Cov}[\eta_{t+n+1}, Z_t] = C$. Recall that the representational power of the PLG comes from this property of the noise term. The value of C does not depend on Z_t; while this might be fine in a linear system, it is easy to construct nonlinear examples where C should vary with Z_t, but rewriting in the dual form has failed to capture

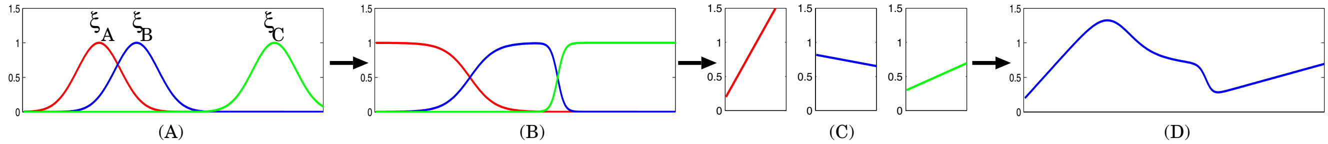

Figure 2: Mixing linear models with renormalized Gaussians. The Gaussians (A) are renormalized (B); each Gaussian is associated with a linear function (C), and then mixed together to generate a final function (D). Note its linear generalization.

this. In contrast, the MPLG uses J noise terms, each with different properties; since these are combined with weights that are a function of Z_t, there is effectively a *composite* noise term which indirectly depends on Z_t.

The kernel autoregressive (KAR) model states that the next observation is a nonlinear function of the *past* n observations z_{t-n}: $\mathrm{E}[Y_{t+1}] = \sum_{j=1}^{J} \alpha_j K(\xi_j, z_{t-n})$. It has a similar form to the KPLG: the same kernels, basis function centers, and coefficients are used, and it can be trained similarly. However, KAR assumes that n past observations constitute state, while the KPLG (and MPLG) can summarize a potentially infinite amount of history into their predictions.

Hybrid Particle-Analytical Inference

We have been discussing how to model Y_{t+n+1} as a function of Z_t, but this is only part of the total state update mechanism. Recall that the state update (Eqs. 2 and 3) requires three terms: $E_t = \mathrm{E}[Y_{t+n+1}]$, $C_t = \mathrm{Cov}[Y_{t+n+1}, Z_t]$ and $V_t = \mathrm{Var}[Y_{t+n+1}]$. As noted, computing E_t, C_t and V_t in closed form is usually impossible. This section therefore discusses sigma-point approximations (or SPAs). After, we discuss another contribution of this paper, which is our general hybrid particle-analytical inference method.

Sigma Point Approximations

Sigma-point approximations (or "unscented transformations"), are a general method of propagating random variables through a nonlinear function (Julier & Uhlmann 1996). The method is a conceptually simple deterministic sampling approach. Suppose we are given a random variable $Y = f(P)$ that is a nonlinear function of a multivariate Gaussian random variable P. Instead of recording P's distribution in terms of a mean and covariance, we represent the same information with a small, carefully chosen number of *sigma points*. These points are selected so that they have the same mean and covariance as P (in fact, they are the minimal such set), but the advantage is that they can be propagated *directly* through the function $f()$. We then compute the posterior statistics of the points to approximate Y.

There are many advantages to SPAs: they are simple and flexible, and are also provably accurate to at least a second order approximation of the dynamics for any distribution on P and any nonlinearity, and are accurate to third order for a Gaussian distribution on P and any nonlinearity. Fourth order terms can sometimes be corrected as well.

There are important differences between SPAs and particle filters. Particle filters typically allow a multi-modal distribution over states, while SPAs require a Gaussian; it is the Gaussian assumption which gives the SPA its strong theoretical guarantees with a small number of points. Also, particle filters sample randomly, but SPAs sample deterministically.

A Hybrid Approach to Inference

Given a k-dimensional multivariate Gaussian, a SPA instantiates $2k$ sigma-points, each of which is propagated through the function $f()$. A naive use of sigma-point approximations in the context of the MPLG would be to construct $2(n + J)$ points, based on the joint Gaussian:

$$P = \begin{pmatrix} Z_t \\ \eta^1_{t+n+1} \\ \vdots \\ \eta^J_{t+n+1} \end{pmatrix} \sim \mathcal{N}\left[\begin{pmatrix} \mu_t \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} \Sigma_t & C^1_\eta & \cdots & C^J_\eta \\ C^{1T}_\eta & \sigma^{21}_\eta & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C^{JT}_\eta & 0 & \cdots & \sigma^{2J}_\eta \end{pmatrix} \right]$$

This is particularly inefficient if J is large (say, hundreds or thousands), because it results in $2(n + J)$ distinct values for Z_t, and thus in $O(J^2)$ kernel evaluations.

There is a much better way, however, which is based on the following crucial observation: that although there are $n + J$ random variables, the nonlinearities on the model are only a function of n of those variables – in particular, the n variables in the vector Z_t. Our contribution is to combine this insight with the smoothing properties of conditional expectations (also called the conditional expectation identity), which in the case of the MPLG states that

$$E_t = \mathrm{E}_Z \left[\mathrm{E}[Y_{t+n+1} | Z_t = z^i] \right]$$

$$C_t = \mathrm{E}_Z \left[\mathrm{E}[Y_{t+n+1} Z_t^T | Z_t = z^i] \right] - \mathrm{E}[Y_{t+n+1}]\mathrm{E}[Z_t^T]$$

$$V_t = \mathrm{E}_Z \left[\mathrm{E}[Y_{t+n+1} Y_{t+n+1}^T | Z_t = z^i] \right] - \mathrm{E}[Y_{t+n+1}]\mathrm{E}[Y_{t+n+1}^T].$$

Our strategy is to combine these facts by *partially* instantiating sigma-points – in particular, we only instantiate $2n$ sigma-points describing Z_t. *Given* those sigma-points, the interior expectations are analytically tractable, and we compute the exterior expectations *over* those sigma-points.

To compute the interior expectations, let $w_i \in \mathbb{R}^{J,1}$ be the weights of the i-th sigma-point $z^{(i)}$ and $G \in \mathbb{R}^{J,n}$ be a matrix whose j-th row is g^{jT}. Define vectors Q and B with $Q_j = \eta^j_{t+n+1}$ and $B_j = b^j$, let $L \in \mathbb{R}^{J,n}$ be a matrix whose j-th row is C^{jT}_η, and let $M \in \mathbb{R}^{J,J}$ be a diagonal matrix where $M_{j,j} = \sigma^{2j}_\eta$. Let $H_i = \mathrm{E}[w_i^T (G z^{(i)} + B) | Z_t = z^{(i)}] = w_i^T (G z^{(i)} + B)$ and $F_i = \mathrm{E}[Q | Z_t = z^{(i)}] = L \Sigma_t^{-1} (z^{(i)} - \mu_t)$. Then:

$$E_t^{(i)} = \mathrm{E}[Y_{t+n+1} | Z_t = z^{(i)}] = H_i + F_i \qquad (9)$$

$$C_t^{(i)} = \mathrm{E}[Y_{t+n+1} Z_t^T | Z_t = z^{(i)}] = (H_i + F_i) z^{iT} \qquad (10)$$

$$V_t^{(i)} = \mathrm{E}[Y^2_{t+n+1} | Z_t = z^{(i)}] = H_i H_i^T + H_i F_i^T + F_i H_i^T + w_i^T (D - \mathrm{diag}(\mathrm{diag}(L \Sigma_t^{-1} L^T)) + F_i F_i^T) w_i \qquad (11)$$

- Construct a set of $2n$ sigma points describing Z_t:
  $z_t^{(2i-1)} = \mu_t + (\sqrt{n\Sigma_t})_i \quad z_t^{(2i)} = \mu_t - (\sqrt{n\Sigma_t})_i$
- Compute a weight vector for each sigma point:
  $w(z_t^{(i)})_j = K(\xi_j, z_t^{(i)}; \phi_j)/(\sum_{k=1}^{J} K(\xi_k, z_t^{(i)}; \phi_k))$
- For each $z_t^{(i)}$, compute $E_t^{(i)}$ (Eq. 9), $C_t^{(i)}$ (Eq. 10) and $V_t^{(i)}$ (Eq. 11).
- Compute empirical posterior statistics:
  $E_t = \frac{1}{N}\sum_{i=1}^{N} E_t^{(i)} \quad C_t = \frac{1}{N}\sum_{i=1}^{N} C_t^{(i)} - E_t\mu_t$
  $V_t = \frac{1}{N}\sum_{i=1}^{N} V_t^{(i)} - E_t^2$

Figure 3: Hybrid particle-analytical MPLG inference.

We compute Eqs. (9)-(11) for each sigma-point, and then use expectations over them to compute the final terms.

The general method is summarized as follows: First, instantiate sigma-points for the minimum number of variables needed to make the model tractable. Second, analytically compute terms based on the model given the sigma-points. Third, compute posterior statistics using expectation smoothing over the sigma-points.

The final algorithm is shown in Figure 3.

Model Estimation

The MPLG requires several parameters: the dimension n of the system, the basis function centers ξ_j and weights α_j, the noise statistics C and σ_η^2, and the covariance matrix ϕ_j.

We are interested in learning the parameters from training data. This data will be given as a set of trajectories from the system, with each trajectory consisting of at least $n+1$ sequential observations. We will slice these trajectories into all possible training pairs (z_i, y_{i+n+1}) where $z_i \in \mathbb{R}^n$ is a vector of n successive observations (representing a noisy sample of some Z_t), and $y_{i+n+1} \in \mathbb{R}$ is the $(n+1)$-th observation (a sample of the corresponding Y_{t+n+1}, or the state extension). We then collect all the pairs into the set S.

Model Order Selection. We must first estimate the order of the model, which includes the system dimension n and the number of basis functions J. For our experiments, we use cross-validation to select parameters from a set of likely candidates. There is nothing unusual about our model or estimation needs, so many other techniques are also suitable.

Finding Basis Function Parameters. Next, we must determine the basis function centers ξ_j and covariance matrices ϕ_j. We used the dictionary-based selection method of Engel et al. (2004). Specifically, we set $\phi_j = \sigma_\phi^2 I$ and then constructed a set of z_i's whose features are almost linearly independent ("almost" is defined by a threshold parameter).

Estimating Individual PLG Parameters. We can now determine the mixing weights for each PLG at any point, so we estimate the parameters of each PLG individually, using weighted versions of the regressions and sample statistics needed. To start, we collect each y_{i+n+1} into a vector $Y \in \mathbb{R}^{|S|}$, and collect each z_i^T into a matrix $Z \in \mathbb{R}^{|S|,n}$. We then compute the weights for each training point using our renormalized kernels: $w(z_i)_j = K(\xi_j, z_i; \phi_j)/(\sum_{j=1}^{J} K(\xi_j, z_i; \phi_j))$. We also define a normalizing constant as the sum of all of the weights for each PLG: $N_j = \sum_{i=1}^{|S|} w(z_i)_j$. For each PLG, collect the weights $w(z_i)_j$ into a diagonal weight matrix $W_j \in \mathbb{R}^{|S|,|S|}$.

Now, we can estimate the linear trend for each PLG j using weighted least-squares: $\widehat{g^j} = (Z^T W_j Z)^{-1} Z^T W_j Y$. To estimate the noise statistics, we first compute the noise term for each training point from the perspective of each PLG, which is $\eta_{ij} = (y_{i+n+1} - \langle \widehat{g^j}, z_i \rangle)$. Then, the estimated variance of η_{t+n+1}^j is $\widehat{\sigma_\eta^{2j}} = \frac{1}{N_j - 1}\sum_{i=1}^{|S|} w(z_i)_j (\eta_{ij})^2$. To estimate C_η^j, we run the algorithm on the training data with $C_\eta^j = 0$ and record our estimate of μ_t at each t, called μ_i. We then compute $\mathrm{Cov}[Z_t, \eta_{t+n+1}] = \mathrm{E}[(Z_t - \mu_t)(\eta_{t+n+1})]$, which is simply $\widehat{C_\eta^j} = \frac{1}{N_j - 1}\sum_{i=1}^{|S|} w(z_i)_j (z_i - \mu_i)\eta_{ij}$. Note that while estimating both σ_η^{2j} and C_η^j we have used the fact that $\mathrm{E}[\eta_{t+n+1}^j] = 0$ for all t.

Estimating KPLG and KAR Parameters. Parameters for the KPLG and KAR algorithms were estimated similarly to those of the MPLG. The ξ_j's were selected with a dictionary, and the α's were computed using regularized least-squares kernel regression, with λ the regularization coefficient.

Experiments and Results

We tested the PLG, MPLG, KPLG and KAR algorithms on one LDS (the "Rotation" problem) and four nonlinear dynamical systems (the "Biped," "Peanut," "NB3," and "Spring" problems), where the underlying generative model was known. Since the models are limited to scalar observations, we also tested on three well-known timeseries benchmarks (Santa Fe Laser, Mackey-Glass, and K.U. Leuven). A sigma-point approximation was used for the KPLG.

Parameters were selected by 10-fold cross-validation. Algorithms were judged on the mean-squared error (MSE) of their predictions, meaning we are not attempting to estimate latent state (but we are allowed to *use* state). All data sets were normalized to be in $[0, 1]$. All algorithms used the Gaussian kernel. For the initial state, we set Σ_0=$1e^{-5}I$ and μ_0 to be the first n values of the test sequence. Algorithms were tested on n=$2, \cdots, 6$, σ_ϕ^2=$0.1, 0.4, 0.8, 1.2$, λ=$0.00001, 0.001, 0.01$, and ν=$0.0001, 0.001, 0.01$ (ν is the dictionary threshold). All problems except Laser were trained on 2000 sequential observations and tested on a 200 observation continuation; Laser had 1000 training and 100 testing observations. **Descriptions:** Rotation, Peanut, Biped, NB3 are 2D dynamical systems. Rotation is linear; the others are nonlinear. Observations and dynamics were noisy. Spring is a 2D system with a mass oscillating between nonlinearly damped springs. Only the position of the mass was observed. Deterministic with noise-free observations. Mackey-Glass is a deterministic but chaotic timeseries gen-

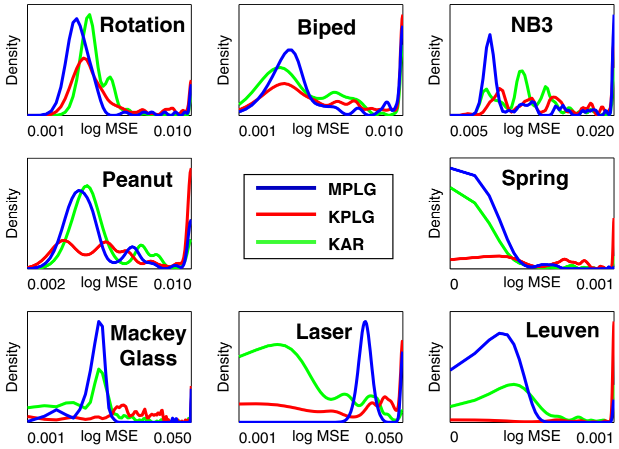

Figure 4: Qualitative comparison of the nonlinear algorithms. Shown is the density of all MSEs generated.

erated from a delay differential equation. Parameters were a=0.2, b=0.1, and τ=30. Laser is data from the Santa Fe timeseries competition. Leuven is competition data from the International Workshop on *Advanced Black-Box Techniques for Nonlinear Modeling*, K.U. Leuven Belgium, 1998.

Results

Figures 4 and 5 summarize our results, which are very encouraging. Figure 4 shows the results qualitatively. Each parameter setting for each algorithm generated a MSE; the figure plots their log distribution. This examines the expected performance for any given parameter setting; a sharply peaked distribution on the left side is desired (implying low expected MSE). Outliers are lumped on the right-hand side.

From Figure 4, three results are evident. First, the MPLG's density curve is often stacked on the left-hand side, as desired. The curve is also peaked, indicating that its performance is insensitive to the exact choice of parameter. It also shows fewer outliers than the KPLG, suggesting that the MPLG is more stable than the KPLG.

Figure 5 shows our results quantitatively. Here, we have used 10-fold cross-validation to select parameters; all algorithms with an MSE within 5% of the lowest are reported as “Best.” The MPLG is among the best performing algorithms on four out of eight problems, and in particular, it performed well on the nonlinear dynamical systems, where there really *is* an opportunity to leverage infinite memory via state. In contrast, KAR has performed well on the timeseries problems; in particular, it wins on the Mackey-Glass series, which really *is* an autoregressive model. The exception is Spring, but this is expected: it is deterministic and noiseless, so n past observations and n predictions are equivalent; the uncertainty the MPLG/KPLG models is unhelpful. PLG won on the linear problem, which is also unsurprising.

Together, the quantitative and qualitative results suggest several conclusions. First, that not only are the very best MSEs often obtained with the MPLG, but for any given parameter setting, the MPLG is likely to outperform other models. Second, that the nonlinear models are outperforming their linear counterparts. Third, that the MPLG is superior to the alternative nonlinear version of the PLG, the KPLG. Fourth, that the MPLG is indeed capturing state, and is therefore superior to autoregressive models in situations where state can be leveraged. Not reflected in these results is the fact that the best parameters were rarely selected for KPLG because of outliers in the cross-validation runs, but this is part of the point: MPLG is more stable than KPLG.

Problem	Best	Problem	Best
Rotation	PLG	Spring	KAR
Biped	MPLG	M.G.	KAR
Peanut	MPLG	Leuven	MPLG/PLG
NB3	MPLG/PLG/KAR	Laser	KAR

Figure 5: Best performing algorithms on the test problems.

Conclusions and Future Research

We have investigated the idea of predictive representations of state to model continuous observation, nonlinear, stochastic dynamical systems, and have proposed a specific mixture model derived from the PLG. Based on our experiments, the most general conclusion is that both the idea and the model are viable. The MPLG has experimentally demonstrated good, stable performance on almost all of the problems tested here. We have also contributed a general hybrid particle-analytical inference method, which is fast, accurate and easy to use, and which makes our model tractable. It improves the utility of sigma-point approximations in general, and could find application in other contexts.

There is still work to be done, but the MPLG learns reasonable models directly from data, and is competitive with other methods. In particular, the MPLG appears to best both the KPLG and simple kernel autoregression. Superiority over KAR suggests that the MPLG is indeed leveraging the advantages of state, meaning the MPLG is another example of a successful predictive representation of state.

Acknowledgments

This work is supported by the National Science Foundation under Grant Number IIS-0413004 and by an NSF Graduate Research Fellowship to David Wingate.

References

Bishop, C. M. 1995. *Neural Networks for Pattern Recognition*. Oxford University Press.

Engel, Y.; Mannor, S.; and Meir, R. 2004. The kernel recursive least squares algorithm. *IEEE Transactions on Signal Processing* 52(8):2275–2285.

Julier, S., and Uhlmann, J. K. 1996. A general method for approximating nonlinear transformations of probability distributions. Technical report, University of Oxford.

Littman, M. L.; Sutton, R. S.; and Singh, S. 2001. Predictive representations of state. In *NIPS 14*, 1555–1561.

Rudary, M. R.; Singh, S.; and Wingate, D. 2005. Predictive linear-Gaussian models of stochastic dynamical systems. In *21st Conference on Uncertainty in Artificial Intelligence*, 501–508.

Wingate, D., and Singh, S. 2006. Kernel predictive linear Gaussian models for nonlinear stochastic dynamical systems. In *International Conference on Machine Learning*.