Learning Time Series Models for Pedestrian Motion Prediction*

Chenghui Zhou[1], Borja Balle[1], and Joelle Pineau[1]

***Abstract*— Robot systems deployed in real-world environments often need to interact with other dynamic objects, such as pedestrians, cars, bicycles or other vehicles. In such cases, it is useful to have a good predictive model of the object's motion to factor in when optimizing the robot's own behaviour. In this paper we consider motion models cast in the Predictive Linear Gaussian (PLG) model, and propose two learning approaches for this framework: one based on the method of moments and the other on a least-squares criteria. We evaluate the approaches on several synthetic datasets, and deploy the system on a wheelchair robot, to improve its ability to follow a walking companion.**

I. INTRODUCTION

Time series data is ubiquitous in robotics, where streams of sensor readings are acquired in real-time, and used by the robot to decide on a course of action. In some cases, the sensor data contains observations of dynamic objects in the environment, and then the robot's decision depend on the motion patterns of the said object. It is particularly useful for the robot to be able to accurately predict the trajectory of the moving object in order to adapt its own course of action. A key challenge here is to obtain a good predictive model of the object's motion.

This paper tackles the problem of inferring a dynamic model directly from time-series data. This contribution is motivated by previous work on building algorithms for robust person following for social and service robots [1], [2], [3], [4], [5], [6]. Smart wheelchairs built in our lab are designed to operate in a variety of human environments, and have a crucial need to adapt their motion behaviour to the presence of pedestrians [6]. This arises when the smart wheelchair is in autonomous navigation mode and wishes to reach a goal while avoiding incoming pedestrians (Figure 1 (left)). Another case is when the wheelchair user is accompanying a walking person (or a group of people), and the wheelchair must adapt its motion to the walking patterns of others (Figure 1 (right)). In such situations, having an accurate predictive motion model of the person (or the group) would be very useful. Up until recently, our system used a linear model with constant position and velocity, defined by hand-specified parameters, to predict behaviours of nearby pedestrians, but we observed that this model lacks flexibility and is often inaccurate.

In short, we aim to develop a system that can automatically learn the motion model of a dynamic object from real-data.

Fig. 1. Smart wheelchair avoiding a collision (left) and moving with companions (right).

It should make fast predictions (to accommodate real-time planning), deal with high-dimensional observations, show robustness when learning from small amount of data, and, ideally, provide a notion of uncertainty over its predictions.

Several methods have been considered for this problem. Auto-regressive moving-average (ARMA) models [7] can be estimated from data using least-squares approaches, but they typically focus on one-dimensional time series with observations corrupted by white noise. An alternative is to consider arbitrary linear dynamical systems (LDS) and estimate parameters via the Expectation-Maximization (EM) algorithm [8]; this approach is reasonably flexible but is limited to learning a locally optimal solution, and as we will see in our results, does not produce very good predictions. Finally, predictive linear gaussian (PLG) models provide a multi-variate extension of ARMA models specialized for gaussian noise [9]. PLGs are appealing for robotics applications because they can tackle multi-variate time series and are optimized for predicting future observations. However, existing learning algorithms for PLG suffer from severe limitations and are only practical for one-dimensional observations.

The primary technical contribution of this paper is to extend methods for learning PLG models by proposing method of moments and least-squares solutions. The algorithms we propose are more robust than the previous PLG learning algorithm and extend easily to the multi-variate case. Using synthetic data, we provide a rigorous experimental comparison between the existing EM for LDS, and the new least-squares and method of moments approaches for PLG. We then apply the least-squares for PLG method to learn the walking behaviours of pedestrians during deployment of our smart wheelchair. Results show good performance across the variety of settings considered. We also show how our framework can incorporate sparsity constraints to help scale

*This project was financially supported by by the Natural Resources and Engineering Research Council Canada (NSERC) through the NSERC Canadian Field Robotics Network (NCFRN) and Discovery program.

[1]School of Computer Science, McGill University, Montréal, Canada. {chenghui.zhou@mail.mcgill.ca, bballe@cs.mcgill.ca, jpineau@cs.mcgill.ca}

learning from high-dimensional observations.

II. Time-Series Models with Gaussian Noise

In this section we introduce the two time series models that will be used throughout the paper. One is the classic linear dynamical system (LDS) [10], the other is the less known predictive linear-gaussian model (PLG) [9]. Both models are characterised by the linearity of their state update and observation generation, and the assumption that the noise distribution is Gaussian. In fact, both models are known to be equivalent up to some natural transformation of their state representation. The fundamental difference between the two models is in how the state is represented: in one the state is a latent vector that evolves over time, while in the other the state is the distribution over future observations parametrised by their mean and covariance [9]. In the two models t denotes a discrete variable indexing time, and the time series is given by a sequence of multi-variate observations $(y_t)_{t\geq 0}$ with $y_t \in \mathbb{R}^m$, where m represents the dimension of the observation vectors.

A. Linear Dynamical Systems

The discrete-time uncontrolled stochastic linear dynamical system (LDS) is a classical model for time series widely used in control theory, signal processing, and econometrics [11]. In an LDS, the observation at time t is a noisy linear transformation $y_t = Ox_t + \delta_t$ of a hidden state vector $x_t \in \mathbb{R}^n$ that evolves over time, and the perturbation follows a multi-variate Gaussian distribution $\delta_t \sim \mathcal{N}(0, \Delta)$. Note that the mean and covariance of δ_t are independent of time – i.e. the observation noise process is stationary. The noises $\delta_t, \delta_{t'}$ at different time steps are assumed to be independent, and in particular $\mathbb{E}[\delta_t \delta_{t'}^\top] = 0$ for $t \neq t'$. The hidden state evolves following linear dynamics perturbed by Gaussian noise: $x_{t+1} = Tx_t + \sigma_t$, where $\sigma_t \sim \mathcal{N}(0, \Sigma)$ and again we assume $\mathbb{E}[\sigma_t \sigma_{t'}^\top] = 0$ for $t \neq t'$. In general, the state and observation noises at the same time t are not necessarily independent; their covariance is denoted by $\Lambda = \mathbb{E}[\sigma_t \delta_t^\top]$. To completely specify the model, an initial setting for the hidden state is required. A usual assumption is that x_0 is drawn from a Gaussian distribution $\mathcal{N}(\mu_0, \Sigma_0)$. Predicting and filtering with an LDS can be done with the classical Kalman filter.

B. Predictive Linear–Gaussian Models

The predictive linear–gaussian (PLG) model was introduced in a series of papers [9], [12] inspired by the success of the predictive state representation for time series with discrete observations in reinforcement learning. Predictive models are characterised by the use of a set of sufficient statistics about the future of the process as their state; see [13] for a general discussion of the principles behind predictive representations. In this paper we focus on the multi-variate uncontrolled PLG presented in [9].

To define a PLG with a state of length n we start by grouping n consecutive observations from the time series $(y_t)_{t\geq 0}$ into a single vector:

$$Y_t = \begin{bmatrix} y_t \\ y_{t+1} \\ \vdots \\ y_{t+n-1} \end{bmatrix} \in \mathbb{R}^{(n\cdot m)} \ .$$

The first assumption of the model is that such vectors follow a joint multi-variate Gaussian distribution. In particular, after observing $y_0, \ldots, y_t$, the state at time t is given by the mean μ_t and the covariance Σ_t of the next n observations: $Y_{t+1} \sim \mathcal{N}(\mu_t, \Sigma_t)$.

The second assumption of the model is that the state can be updated linearly once y_{t+1} is given. This can also be formulated in terms of the distribution of y_{t+n} conditioned on Y_t. In particular, we assume that $y_{t+n}|Y_t \sim \mathcal{N}(GY_t, \Delta)$, which can also be written as $y_{t+n}|Y_t = GY_t + \delta_{t+n}$, where $\delta_{t+n}|Y_t \sim \mathcal{N}(0, \Delta)$. Here $G \in \mathbb{R}^{m\times(n\cdot m)}$ is a *mean update matrix* and $\Delta \in \mathbb{R}^{m\times m}$ is a noise covariance matrix. We note that Δ is the covariance of the noise δ_{t+n} conditioned on Y_t; however, Y_t and δ_{t+n} are not in general independent, and we write $C = \mathbb{E}[Y_t \delta_{t+n}^\top] \in \mathbb{R}^{(n\cdot m)\times m}$ for their covariance. To complete the specification of a PLG model one needs to specify the initial mean μ_0 and covariance Σ_0 of the first n observations in Y_0.

In order to derive the state update equations for the PLG we shall introduce the following notation. Let $\Delta', C', G' \in \mathbb{R}^{(n\cdot m)\times(n\cdot m)}$ be the block-matrices given by

$$\Delta' = \begin{bmatrix} 0 & 0 \\ 0 & \Delta \end{bmatrix} \ , \ C' = \begin{bmatrix} 0 & C \end{bmatrix} \ , \ G' = \left[\begin{array}{cc} 0 & I \\ \hline \multicolumn{2}{c}{G} \end{array}\right] \ .$$

Let us write $J = [I \ 0] \in \mathbb{R}^{m\times(n\cdot m)}$ and $F_t = (G'\Sigma_t + C'^\top)J^\top$. Now we can use the two assumptions of the PLG model to write the joint distribution of Y_t and y_{t+n} as a multi-variate Gaussian distribution with mean μ_t' and covariance Σ_t'. An important observation is that this distribution is also the joint distribution of y_t and Y_{t+1}. In particular, we have

$$\mu_t' = \begin{bmatrix} J\mu_t \\ G'\mu_t \end{bmatrix} \ ,$$
$$\Sigma_t' = \begin{bmatrix} J\Sigma_t J^\top & F_t^\top \\ F_t & G'\Sigma_t G'^\top + \Delta' + G'C' + C'^\top G'^\top \end{bmatrix} \ .$$

Using this distribution and the formula for the conditional distribution of a multi-variate Gaussian given some of its coordinates we obtain the following state update equations describing the distribution of Y_{t+1} given y_t:

$$\begin{aligned} \mu_{t+1}|y_t &= G'\mu_t + F_t(J\Sigma_t J^\top)^{-1}(y_t - J\mu_t) \ , \\ \Sigma_{t+1} &= G'\Sigma_t G'^\top + \Delta' + G'C' \\ &\quad + C'^\top G'^\top - F_t(J\Sigma_t J^\top)^{-1}F_t^\top \ . \end{aligned}$$

III. Learning Algorithms

The learning setup we consider assumes there is a fixed stochastic process capable of generating independent realisations of the same time series, and we can use many of these realisations in order to estimate the target model.

More precisely, for $1 \leq i \leq N$ we assume that $\mathbf{y}^{(i)} = (y_0^{(i)}, y_1^{(i)}, \dots y_T^{(i)})$ are independent identically distributed time series of length T drawn from the same process. We use the data in $(\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(N)})$ to estimate the parameters of the models. We choose this learning setup because it is the one we encounter in practice for the smart wheelchair application that attracted our attention to the problem considered in this paper; minor variants (e.g. data coming from a single trajectory) could be accommodated without difficulty.

We now review existing methods proposed to estimate parameters in the LDS and PLG models (Sec. III-A and III-B). We then proceed to describe our two proposed algorithms for learning the state update matrix G in a PLG model: one based on a moment matching approach (Sec. III-C), and another based on least-squares regression (Sec. III-D and III-E). The methods differ in how they estimate the state update matrix (G); once that is done, both algorithms recover the noise covariances in the same way, which is described in Sec III-F.

A. Learning LDS via Expectation–Maximization

Most applications of LDS assume that the parameters are designed by experts. For those cases where learning is required, the standard approach is to use Expectation–Maximisation (EM), which implements an iterative heuristic to maximise the likelihood of the training data under the estimated model. The EM algorithm was first proposed for hidden Markov models [14], [15], and latter adapted to many other latent variable models with parametric observation distributions like LDS [8], [16]. In general, EM algorithms are guaranteed to converge to a local maxima of the likelihood function, and several re-starting heuristics can be considered to improve the chances of finding a global maxima. A full review of the algorithm is beyond the scope of this paper, and we assume readers are familiar with the method.

Note that subspace identification methods can also be used to learn LDS from data [17]. However, the approach followed by these methods assumes that the training data comes in the form of one long uninterrupted trajectory. Since this is not the setting arising in our applications, we shall not consider this family of methods in the present paper.

B. Learning PLG: Previous approach

For PLG, the first learning methods were proposed in [9] for the univariate case. Rudary et al. compared their methods with EM for LDS, showing improved likelihood on the training data. However our attempts to replicate these results in the multi-variate case failed, mainly the learning algorithm from [9] is specifically designed for the univariate setting with a fixed initial state (see later in Fig. 2). This lead us to design two new learning algorithms for multi-variate PLG, which are presented in this section. In particular, the second of these algorithms follows the same principle as the algorithm in [9]: the method of moments. This is a common inference principle in statistics and econometrics [18] that was initially proposed by Pearson [19]. It is based on finding equations that relate the parameters of a certain model to its observed moments, and then deriving a learning algorithm by solving for the parameters using empirical moments. The method of moments has seen a renewed interest in the machine learning community in the last few years due to its high computational efficiency [20].

C. Learning PLG via Method of Moments

Our first PLG learning algorithm is based on finding a G that relates two covariance matrices of observations, and can be interpreted as a method of moments because it finds the parameters that best explain the relation between several (empirical) moments associated with the PLG model.

To describe the algorithm we start by introducing a backward history vector

$$B_t = \begin{bmatrix} y_{t-l} \\ \vdots \\ y_{t-1} \end{bmatrix} \in \mathbb{R}^{(m \cdot l)} ,$$

for some length $l \geq 1$. Note that from this point of view, Y_t can be interpreted as a forward history vector at time t. Using B_t and Y_t we can define a covariance matrix at time t that captures the relation between past and future observations:

$$\Sigma_{Y_t,B_t} = \mathbb{E}[Y_t B_t^\top] \in \mathbb{R}^{(n \cdot m) \times (m \cdot l)} .$$

Similarly, we define a covariance matrix between the past from time t and the observation y_{t+n} that is generated immediately after Y_t:

$$\Sigma_{y_{t+n},B_t} = \mathbb{E}[y_{t+n} B_t^\top] \in \mathbb{R}^{m \times (m \cdot l)} .$$

Given the training data $(\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(N)})$, these covariance matrices are estimated as follows:

$$\hat{\Sigma}_{Y_t,B_t} = \frac{1}{N} \sum_{i=1}^{N} Y_t^{(i)} B_t^{(i)\top} ,$$

$$\hat{\Sigma}_{y_{t+n},B_t} = \frac{1}{N} \sum_{i=1}^{N} y_{t+n}^{(i)} B_t^{(i)\top} .$$

Note that these matrices are only defined for $l \leq t \leq T - n$.

Using the notation we just defined, the moment-matching learning algorithm is given by

$$\hat{G}_{\mathrm{MM}} = \left(\sum_{t=l}^{T-n} \hat{\Sigma}_{y_{t+n},B_t} \right) \left(\sum_{t=l}^{T-n} \hat{\Sigma}_{Y_{t+n},B_t} \right)^{\dagger_k} ,$$

where we use $A^{\dagger_k}$ to denote the pseudo-inverse of the best rank k approximation of A given by SVD[1]. The rank k is a regularisation parameter to avoid overfitting (see the discussion below); in practice, the optimal k can be selected by cross-validation. Its purpose is to avoid a blow-up in the estimation noise present in $\sum_{t=l}^{T-n} \hat{\Sigma}_{Y_{t+n},B_t}$ due to the pseudo-inverse operation.

[1]If the full SVD is $A = USV^\top$, the best rank k approximation to A is obtained as $A_k = U_k S_k V_k^\top$, where U_k, V_k contain the first k columns of U and V respectively, and S_k contains the top k singular values of A. Thus, by properties of the pseudo-inverse we have $A^{\dagger_k} = V_k S_k^{-1} U_k^\top$.

In order to understand the motivation behind this learning algorithm one can compute an expression for Σ_{y_{t+n},B_t} as follows:

$$\begin{aligned}\Sigma_{y_{t+n},B_t} &= \mathbb{E}[y_{t+n}B_t^\top] = \mathbb{E}[(GY_t + \delta_{t+n})B_t^\top] \\ &= G\mathbb{E}[Y_t B_t^\top] + \mathbb{E}[\delta_{t+n}]\mathbb{E}[B_t^\top] = G\Sigma_{Y_t,B_t} \ ,\end{aligned}$$

where we used linearity of expectation, that B_t and δ_{t+n} are independent, and that $\mathbb{E}[\delta_{t+n}] = 0$. This relation explains how G can be recovered from Σ_{y_{t+n},B_t} and Σ_{Y_t,B_t}. Since the relation is the same for all t, we decide to average the different covariance estimates in order to obtain a better solution for $\hat{G}_{\mathrm{MM}}$. The choice of l and k is typically done by cross-validation, but at least one should have $k \leq l$. In addition, the bigger l is, the more likely it is that Σ_{Y_t,B_t} will contain enough information to recover G properly, i.e. to ensure that the system of linear equations solved by this method is not underdetermined.

D. Learning PLG via Least-Squares Regression

Our second PLG learning algorithm is a simple least-squares regression which proceeds by computing the minimiser $\hat{G}_{\mathrm{LS}}$ of

$$\min_G \frac{1}{N\cdot(T-n+1)} \sum_{i=1}^{N} \sum_{t=0}^{T-n} \|y_{t+n}^{(i)} - GY_t^{(i)}\|_2 \ .$$

By using the well-known closed-form solution to least-squares problems we can write a direct solution for G without the need to perform any optimisation as follows. For $1 \leq i \leq K$ we define the block-Hankel matrix

$$H^{(i)} = \begin{bmatrix} y_0^{(i)} & \cdots & y_{T-n}^{(i)} \\ \vdots & \ddots & \vdots \\ y_{n-1}^{(i)} & \cdots & y_{T-1}^{(i)} \end{bmatrix} \in \mathbb{R}^{(n\cdot m)\times(T-n+1)} \ .$$

Using these we define the matrix $H = [H^{(1)} \cdots H^{(N)}]$. Similarly, we define the matrix $Y = [y_n^{(1)} \cdots y_T^{(1)} \cdots y_n^{(N)} \cdots y_T^{(N)}]$. With this notation the solution $\hat{G}_{\mathrm{LS}}$ to the least-squares problem above can be obtained as

$$\hat{G}_{\mathrm{LS}} = YH^\dagger \ ,$$

where $H^\dagger = H^\top(HH^\top)^{-1}$ denotes the Moore–Penrose pseudo-inverse.

The rationale behind this learning algorithm is clear: try to find a mean update matrix G that best predicts the next observation given the past n observations as specified by the PLG model. We will see in our experiments that in general this learning algorithm performs quite well with small amounts of data. On the other hand, for large amounts of data the algorithm is biased – due to the use of dependent[2] samples to estimate G – and is outperformed by the method of moments presented above.

[2]Note that although the different series $\mathbf{y}^{(i)}$ are mutually independent, the observations within a single sequence are dependent, leading to a least-squares estimation with dependent observations.

E. Least-Squares Regression for PLG with Sparsity

In some cases it is possible that learning the mean update matrix G directly with the least-squares criteria leads to overfitting, especially when the number of parameters is large in comparison to the amount of available training data. The traditional approach in machine learning to remedy such situation is to include a regularisation term in the least-squares optimisation to help prevent overfitting by trading off training accuracy for model complexity [16]. One such approach is to use a ℓ_1-type penalty inducing sparsity in the set of parameters to be learned. In the case of a PLG, a sparse G can be interpreted as saying that every entry in the observation y_{t+n} depends only on a few entries from Y_t. Following this approach, we obtain a learning algorithm based on computing the minimiser $\hat{G}_{\mathrm{SP}}$ of

$$\min_G \frac{1}{N\cdot(T-n+1)} \sum_{i=1}^{N} \sum_{t=0}^{T-n} \|y_{t+n}^{(i)} - GY_t^{(i)}\|_2 + \lambda\|G\|_{1,1} \ ,$$

where $\lambda > 0$ is a regularisation parameter and $\|G\|_{1,1} = \sum_{i,j} |G_{i,j}|$ is the ℓ_1 norm over all the entries in G. This optimisation problem can be easily reduced to the Lasso [21] and then solved using commonly available implementations [22]. This extension of the least-squares method is particularly appropriate for learning high-dimensional systems from limited data.

F. Estimating the Covariances

Assuming an estimate $\hat{G}$ for G is known, we finally describe how to obtain the remaining parameters of the model. The initial state distribution is straightforward:

$$\hat{\mu}_0 = \frac{1}{N}\sum_{i=1}^{N} Y_0^{(i)} \ ,$$

$$\hat{\Sigma}_0 = \frac{1}{N}\sum_{i=1}^{N} (Y_0^{(i)} - \hat{\mu}_0)(Y_0^{(i)} - \hat{\mu}_0)^\top \ .$$

In order to estimate the covariances related to the noise, one first needs to estimate the noise vectors. Thus, we define $\hat{\delta}_{t+n}^{(i)} = y_{t+n}^{(i)} - \hat{G}Y_t^{(i)}$ for $1 \leq i \leq N$ and $0 \leq t \leq T-n$. These noise estimates can be used to approximate the noise and the noise-state covariances:

$$\hat{\Delta} = \frac{1}{N(T-n+1)} \sum_{i=1}^{N}\sum_{t=0}^{T-n} \hat{\delta}_{t+n}^{(i)}(\hat{\delta}_{t+n}^{(i)})^\top \ ,$$

$$\hat{C} = \frac{1}{N(T-n+1)} \sum_{i=1}^{N}\sum_{t=0}^{T-n} Y_t^{(i)}(\hat{\delta}_{t+n}^{(i)})^\top \ .$$

IV. EMPIRICAL COMPARISON

We consider several synthetic cases to carefully examine the empirical performance of each approach. We then present results with data collected onboard our smart wheelchair robot with real pedestrians.

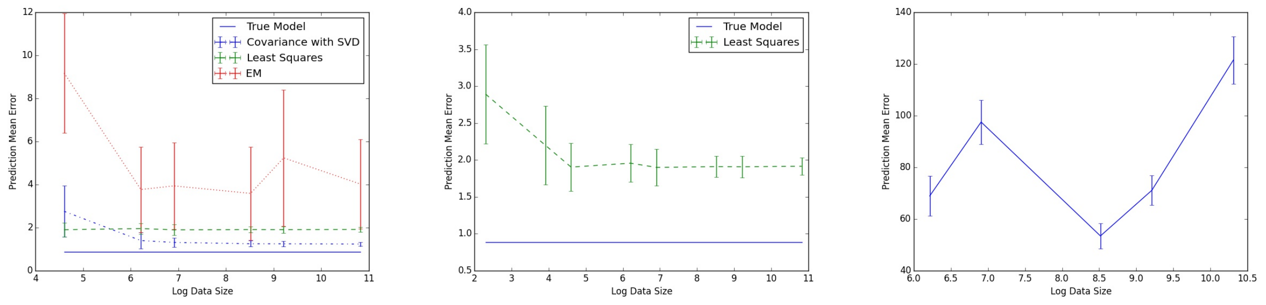

Fig. 2. (left) Comparison of learning methods on the synthetic linear model. (center) Results for PLG learning with least-squares including training sets of size 10 and 50. (right) Results for PLG learning using Rudary et al. [9] original learning method.

A. Synthetic Data: A Simple Linear System

We generated synthetic data from the PLG system equivalent to the linear dynamical system with update and observation matrices

$$T = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad O = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} .$$

Noise covariances are set to $\Delta = 0.09 \cdot I_2$ and $\Sigma = 0.09 \cdot I_4$. The initial state is fixed to $[2, 1, 0.2, 0.2]^\top$. This represents a 2D constant velocity model with hidden state containing position and velocity across the x and y axis: $x_t = [p_x, p_y, v_x, v_y]^\top$. It is easy to see that the observations correspond to the x and y position of the state.

We compare the following methods: PLG learning with methods of moment (labeled: covariance with SVD), PLG learning with the least-square method, as well as LDS learning via the EM algorithm. Learning for each method is done using 100, 500, 1000, 5000, 10,000 and 50,000 sequences. Each sequence has length 30.

Results are shown in Figure 2 (left). The x-axis of the graph is the log sample size. To calculate the mean error, we take the distance of the predicted position and the actual position and average it over the length of the test sequence, which is 100, as well as over all sequences. The variance plotted is calculated by adding the squared distance between actual positions and their corresponding predicted positions and averaging the sum over the number of predictions. Our results show that both PLG learning methods easily outperform the EM algorithm. The PLG learning via least-squares converges fastest to a reasonably good solution. The PLG learning via SVD is able to reach a slightly better result as the number of sequences in the data set increases. The least-squares method appears to be flat over all training sample sizes. However, if we decrease the sample size even further, we can in fact observe the decreasing trend in the leftmost portion of Figure 2 (center) (we do not include the other methods on this graph as their performance is very poor in the small data regime).

Figure 2 (right) shows the performance of the original learning method from Rudary et al. [9]. Note the instability and large prediction error (y-axis). While the method performs well for one-dimensional systems, this clearly does not extend to multi-dimensional cases, thus motivating our development of least-squares and method-of-moments approaches.

We can also measure performance in terms of model likelihood, where better models score higher. Results in Figure 3 (left) follow a similar trend as in Figure 2 (left); the discrimination between the two PLG learning methods is less obvious here, though the SVD method has slightly higher likelihood with more data.

Next, we consider longer prediction horizons, showing the prediction error 2 steps ahead (Fig. 3, center) and 5 steps ahead (Fig. 3, right) – whereas graphs presented above considered only the next step prediction. Results follow the same trend as those of Figure 2 (left), with an advantage again for the least-squares method when there is little data, better performance of the SVD method with more data, and consistently poorer performance with EM. Note that the y-axis is not the same for the different prediction horizons; predictions get consistently worse for all methods with increased prediction horizon.

Next, we consider the same update matrix and observation matrix as above, but change the two noise covariances' diagonal entries from 0.09 to 0.0001 and the basic experiment. Results are presented in Figure 4 (left). The performance of EM, relative to the others, improves significantly. However least-squares still converges fastest to a very good solution, with the SVD learner reaching similar performance with significantly more data.

We repeated our experimental protocol on another model with different parameterisation. Parameters are chosen so that the system is stable (spectral radius of T is < 1), yet the observations and states have non-zero cross-covariances. The

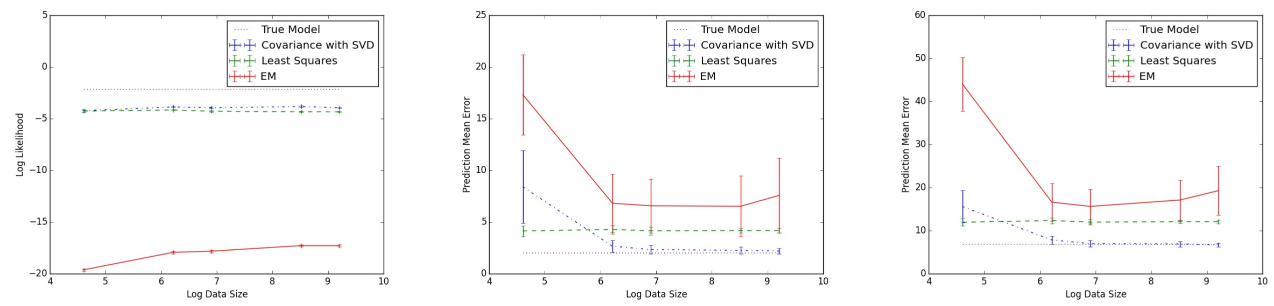

Fig. 3. (left) Comparison of likelihood of learned models on the synthetic linear model. (center) Comparison of prediction accuracy for 2 steps ahead on the synthetic linear model. (right) Comparison of prediction accuracy for 5 steps ahead on the synthetic linear model.

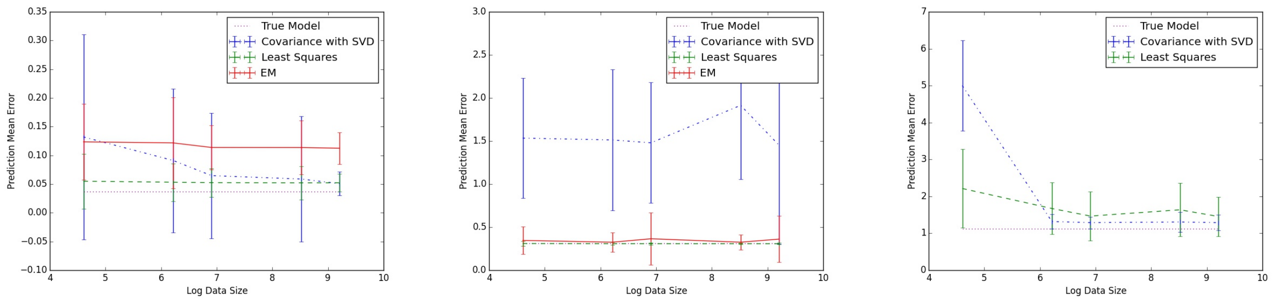

Fig. 4. (left) Comparison of learning methods on a synthetic linear model with smaller noise. (center) Comparison of learning methods on more complex synthetic model. (right) Learning methods comparison on six state dimension synthetic data experiment.

parameters are as follows:

$$T = \begin{bmatrix} 0.8 & 0 & 1.1 & 0 \\ 0 & 0.3 & 0 & 0.7 \\ 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$O = \begin{bmatrix} 1.1 & 0 & 0.5 & 0 \\ 0 & 1.5 & 0 & 0.5 \end{bmatrix}$$

$$\Sigma = 10^{-3} \cdot \begin{bmatrix} 10.3 & 2.2 & 2.2 & 2.2 \\ 2.2 & 10.3 & 2.2 & 2.2 \\ 2.2 & 2.2 & 10.3 & 2.2 \\ 2.2 & 2.2 & 2.2 & 10.3 \end{bmatrix}$$

$$\Delta = 10^{-3} \cdot \begin{bmatrix} 10.1 & 2 \\ 2 & 10.1 \end{bmatrix} .$$

The result of this experiment is plotted in Figure 4 (center). The trends are quite different here. While the least-squares learning maintains good performance, we observe that EM achieves similarly good performance albeit with higher variance, while the PLG learning with SVD is unable to converge to a good solution.

We also considered a system with state dimension 6 and observation dimension 3 given by

$$T = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$O = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} .$$

The state and observation noise covariances are again diagonal with 0.09 variances, and the initial state has mean $[0.3, -0.1, 0.4, 0.02, 0.01, -0.04]^\top$ and diagonal covariance with 0.1 variances. Experiments consider data sets containing 100, 500, 1000, 5000 and 10000 trajectories. We sample 10 data sets for each of the sample sizes. The length of each sequence in the training set is 48. The validation set and test set each contain 1000 sequences of length 70. We do not include performance of the EM method for this larger model because EM failed to converge on this domain. In Figure 4 (left) we compare both PLG learning methods, which show similar trends as above. In Figure 5 (left) we include for the first time the PLG least-squares learning with the sparsity constraint[3] defined in Sec. III-E. We observe that

[3]For each training set size, we used a validation set to select the regularization parameter, and plot results obtained on the test set using that regularization parameter.

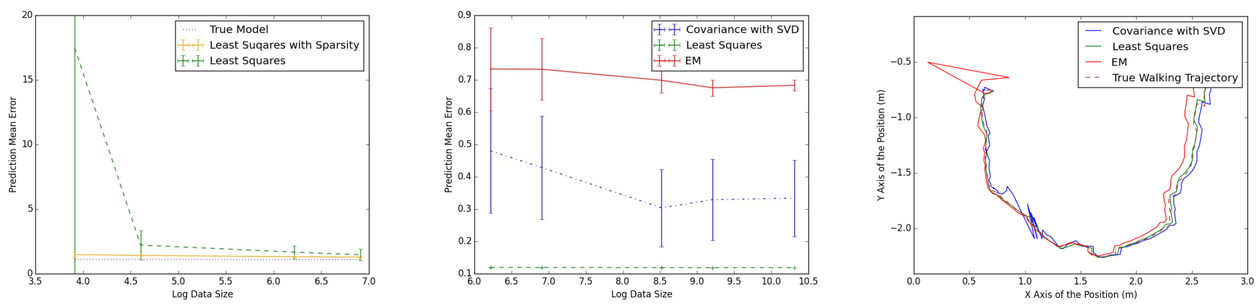

Fig. 5. (left) Comparison between least-squares method with and without sparsity under the 6 dimensional synthetic data model. (center) Result on non-linear synthetic data experiment. (right) Comparison of a true walking trajectory with predictions from learned models.

it offers better stability compared to the standard PLG with least-squares. Note that the results in Figure 4 (left) consider smaller training sets of size 50, 100, 500 and 1000, where the advantage of the sparsity constraint is particularly notable.

B. *Synthetic Data: Non-Linear Systems*

To test robustness of our approach to data generated by a non-linear system we consider the 5-dimensional vehicle re-entry problem described in Sec. 4 of Julier & Uhlmann [23]. We refer the reader to the original publication for the true model parameters. We ran experiments for sample sizes of 500, 1000, 5000, 10000 and 30000 sequences. Each sequence is of length 30. We generated ten training set for each of the sample sizes. For each of the training set, we learn models with state dimensions from 5 up to 11 due to the non-linearity. The best prediction mean error among all the state dimensions learnt is used for the plots. Result of this nonlinear synthetic data experiment are shown in Figure 5 (center). The EM method and the SVD method both show some trend towards improving as the amount of data increases, with SVD outperforming EM for all sample sizes. But the most salient finding is the solid performance of the least-squares method, which has the most stable performance and best prediction for all sizes of training sets.

C. *Real Data: Person Tracking*

Finally, we present results of experiments with real data involving a 2-dimensional time series. At each time step we collect the x and y position of a walking pedestrian. These positions are measured using laser range-finders onboard the robot smart wheelchair platform. Two subjects were recruited to walk near the robot; each was instructed to perform two types of trajectory: straight to simulate usual pedestrian traffic, and curvy to simulate pedestrians avoiding the wheelchair.

All collected data was combined into the training set after some pre-processing. During data collection, pedestrians sometimes walked outside the range of the laser sensors. Some of the sequences can be as long as a few thousands time steps, while others are much shorter. To form our training data set, we split long sequences into segments of at most 30 consecutive time steps, and remove all sequences shorter

TABLE I
EMPIRICAL RESULTS FOR REAL DATA

Type of Model	Prediction Mean Error	Variance
LDS with EM(8)	0.195	0.081
PLG with SVD (5)	0.187	5.96
PLG with SVD (6)	0.248	0.214
PLG with least-squares (10)	**0.118**	**0.037**

than two time steps. In total, we obtain about 5000 training sequences. Since the model state dimension is unknown, we learn multiple models with different state dimensions. The best prediction mean error among all learned models with different state dimensions for each learning method is presented in Table I, along with the specific state dimension selected (shown in brackets beside the method name). Figure 5 (right) shows a real pedestrian trajectory and the predictions produced by three different models. We observe that least-squares produces the smoothest and more accurate prediction, with EM behaving erratically at the beginning of the sequence and SVD producing large oscillations after the pedestrian takes a mildly sharp turn.

For this fixed amount of real data, results show that the least-squares method has the least mean error and variance. The EM method performs somewhat worse than least-squares learning in terms of error and variance. We observe a trade-off between error and variance with the SVD method. This trade-off was not observed in the other two algorithms.

V. CONCLUSIONS

The work presented in this paper tackles the problem of learning a predictive model for dynamical systems. While the use of predictive motion models goes far back in the literature, the set of learning algorithms is rather limited, and so far dominated by Expectation-Maximization-type of approaches. In this paper we consider the recent framework of Predictive Learning Gaussian (PLG) model, and propose two learning algorithms for this model, one based on the methods of moment SVD criteria, and the other based on a least-squares criteria. It is perhaps most interesting to note that while the LDS and PLG models can be shown to be equivalent in terms of dynamics, they lead to important differences in terms of learning algorithms. Throughout several experiments, we observed that the PLG with least-squares approach consistently learned quickly from small amounts of data and provided robust performance to several challenges including varying noise, non-linearity, and higher-dimensions. It is worth noting that the least-squares method gives a biased estimate of the model, so in the case where a lot of data is available, better performance can be obtained with the SVD-based algorithm. Finally, while the work focused on learning predictive models of pedestrians for assistive robots, the methods presented can be applied across several other domains requiring predicting the trajectory of a moving object.

ACKNOWLEDGMENT

We thank Angus Leigh for his help in collecting the data for the real person tracking experiment.

REFERENCES

[1] E. A. Topp and H. I. Christensen, "Tracking for following and passing persons." in *IROS*, 2005.

[2] R. Gockley, J. Forlizzi, and R. Simmons, "Natural person-following behavior for social robots," in *HRI*, 2007.

[3] K. Arras, B. Lau, S. Grzonka, M. Luber, O. Mozos, D. Meyer-Delius, and W. Burgard, "Range-based people detection and tracking for socially enabled service robots," in *Towards Service Robots for Everyday Environments*, 2012, pp. 235–280.

[4] D. V. Lu and W. D. Smart, "Towards more efficient navigation for robots and humans," in *IROS*, 2013.

[5] A. Cosgun, D. A. Florencio, and H. I. Christensen, "Autonomous person following for telepresence robots," in *ICRA*, 2013.

[6] A. Leigh, J. Pineau, N. Olmedo, and H. Zhang, "Person tracking and following with 2d laser scanners," in *ICRA*, 2015.

[7] W. Fuller, *Introduction to statistical time series*. Wiley, 2009.

[8] Z. Ghahramani and G. Hinton, "Parameter estimation for linear dynamical systems," U. of Toronto, Tech. Rep. CRG-TR-96-2, 1996.

[9] M. Rudary, S. Singh, and D. Wingate, "Predictive linear-gaussian models of stochastic dynamic systems," in *UAI 21*, 2005.

[10] S. Roweis and Z. Ghahramani, "A unifying review of linear gaussian models," *Neural computation*, vol. 11, no. 2, pp. 305–345, 1999.

[11] R. Kalman, "Mathematical description of linear dynamical systems," *J. Society for Industrial & Applied Mathematics*, 1963.

[12] S. Singh, M. James, and M. Rudary, "Predictive state representations: A new theory fo rmodeling dynamical systems," in *UAI 20*, 2004.

[13] M. Littman, R. Sutton, and S. Singh, "Predictive representations of state," in *NIPS 14*, 2001.

[14] L. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *Annals of Mathematical Statistics*, vol. 37, no. 6, pp. 1554–1563, 1966.

[15] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proc. of the IEEE*, vol. 77, no. 2, 1989.

[16] C. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[17] P. Van Overschee and B. De Moor, *Subspace identification for linear systems: Theory, Implementation, Applications*. Springer Science & Business Media, 2012.

[18] A. R. Hall, *Generalized method of moments*. Oxford University Press Oxford, 2005.

[19] K. Pearson, "Contributions to the mathematical theory of evolution," *Philosophical Transactions of the Royal Society of London. A*, pp. 71–110, 1894.

[20] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky, "Tensor decompositions for learning latent variable models," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 2773–2832, 2014.

[21] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. of the Royal Statistics Society B*, vol. 58, no. 1, pp. 267–288, 1996.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[23] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," in *AeroSense'97*. International Society for Optics and Photonics, 1997, pp. 182–193.