Towards Spatiotemporal SSM via Linear Dynamical Systems

Jadie Adams
CS6190 Project
GitHub Repo: `https://github.com/jadie1/LDS-EM`

1 Introduction

Background: Statistical shape modeling (SSM) is a useful clinical tool for capturing and quantifying variation of anatomical shape across a population. An effective representations of shape in SSM are landmarks - sets of points consistently defined on the shape surface such that they are in correspondence across the population [5]. ShapeWorks [2, 3] is an open source software developed by my research group, that allows for automatically placing dense sets of landmarks or correspondence points on shapes segmented from 3D medical images, an example of which can be seen in Figure 1. Automatic point placement is done via an entropy-based optimization scheme that balances two goals – the first is maximal correspondence between points across shapes (i.e. smaller model complexity) and the second is maximal uniform spread of points across individual shapes so that the shape is well-captured (i.e. better geometric accuracy).

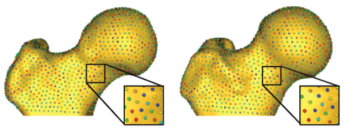


Figure 1: Example of a shape model of two partial femur bones where color denotes correspondence.

Motivation: Many questions that arise about shape in a clinical setting have some dependency on time, i.e. involve spatiotemporal data. For example, we may want to quantify how the shape of the left atrium of the heart varies in relation to the population over the cardiac cycle. Currently ShapeWorks and other computational SSM techniques do not support spatiotemporal data as there is no way to capture the time-dependency. This could be addressed by adding a probabilistic time-series model. A probabilistic approach is ideal here because the generative framework provides a solution for the cases where time-points are missing or we have variable sequence length.

Goal: In this project, I intend to make strides toward spatiotemporal SSM by considering a probabilistic model that can effectively capture sequences of correspondence points. If this is successful, such a model could be integrated into the ShapeWorks optimization scheme, by computing the entropy of the data with respect to the time-series probabilistic model. Points could then be moved in part so that they reduce this entropy, better capturing the true time dependency.

2 Methods

To achieve this goal I will fit a **linear dynamical system (LDS)** to example data for which we know the true time-dependency. I will then evaluate how accurate this model is in terms of generalization, specificity, and inference.

2.1 Notation

In this work, a dataset is comprised of a cohort of N shape sequences denoted $\mathcal{X}$. A given individual observation, $\mathbf{X}_n \in \mathcal{X}$ where $(n = 1, 2, \ldots, N)$, has length T_n and is defined as: $\mathbf{X}_n = \{\mathbf{x}_{n,1}, \mathbf{x}_{n,2}, \ldots, \mathbf{x}_{n,T_n}\} = \{\mathbf{x}_{n,t}\}_{t=1}^{T_n}$. Each $\mathbf{x}_{n,t}$ is a set of M correspondence points which have d dimensions, $\mathbf{x}_{n,t} = \left[\mathbf{x}_{n,t}^1, \mathbf{x}_{n,t}^2, \ldots, \mathbf{x}_{n,t}^M\right] \in \mathbb{R}^{dM}$ $(\mathbf{x}_{n,t}^m \in \mathbb{R}^d)$.So $\mathbf{x}_{n,t}^m$ is the vector of the $\{x, y, z\}$ coordinates of the m^{th} point at the t^{th} time point for the n^{th} individual in the cohort. Now given the observations $\mathcal{X}$, we define corresponding hidden states $\mathcal{Z}$ where for each $\mathbf{Z}_n \in \mathcal{Z}$ where $(n = 1, 2, \ldots, N)$: $\mathbf{Z}_n = \{\mathbf{z}_{n,1}, \mathbf{z}_{n,2}, \ldots, \mathbf{z}_{n,T_n}\} = \{\mathbf{z}_{n,t}\}_{t=1}^{T_n}$. Where $\mathbf{z}_{n,t}$ is the L dimensional latent representation of the particle set for the n^{th} at time t $(\mathbf{z}_{n,t} \in \mathbb{R}^L)$.

2.2 Time-Variant Linear Dynamical System

A linear dynamical system (LDS) is a state space model which models linear transitions and has a Gaussian latent representation (Figure 2). It is defined by the following equations [6, 1].

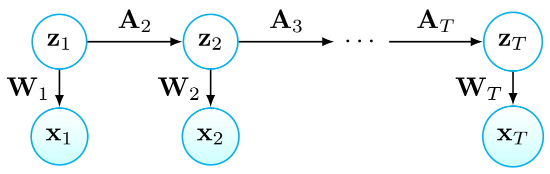
$$\text{State Equation:} \quad \mathbf{z}_{n,t} = \mathbf{A}_t \mathbf{z}_{n,t-1} + \boldsymbol{\epsilon}_{n,t}^z \quad \text{where } \boldsymbol{\epsilon}_{n,t}^z \in \mathcal{N}(0, \boldsymbol{\Sigma}^z) \tag{1}$$

$$\text{Observation Equation:} \quad \mathbf{x}_{n,t} = \mathbf{W}_t \mathbf{z}_{n,t} + \boldsymbol{\epsilon}_{n,t}^x \quad \text{where } \boldsymbol{\epsilon}_{n,t}^x \in \mathcal{N}(0, \boldsymbol{\Sigma}^x) \tag{2}$$

$$\text{Prior Equation:} \quad \mathbf{z}_{n,1} = \boldsymbol{\mu}_0 + \boldsymbol{\varphi}_{n,0} \quad \text{where } \boldsymbol{\varphi}_{n,0} \in \mathcal{N}(0, \mathbf{V}_0) \tag{3}$$

Here $\mathbf{z}_{n,t} \in \mathbb{R}^L$ state (latent) vector, $\mathbf{A}_t \in \mathbb{R}^{L \times L}$ is the transition matrix, $\boldsymbol{\Sigma}^z \in \mathbb{R}^{L \times L}$ is the state covariance matrix, $\mathbf{W}_t \in \mathbb{R}^{dM \times L}$ is the loading/observation-system matrix, $\boldsymbol{\Sigma}^x \in \mathbb{R}^{dM \times dM}$ is the observation covariance matrix, $\boldsymbol{\mu}_0 \in \mathbb{R}^L$ is the latent prior mean, and $\mathbf{V}_0 \in \mathbb{R}^{L \times L}$ is the latent prior covariance matrix.

Thus the time-variant LDS parameters are: $\boldsymbol{\theta} = \{\{\mathbf{A}_t, \mathbf{W}_t\}_{t=1}^T, \boldsymbol{\Sigma}^z, \boldsymbol{\Sigma}^x, \boldsymbol{\mu}_0, \mathbf{V}_0\}$



$$\begin{tikzcd} \mathbf{z}_1 \arrow[r, "\mathbf{A}_2"] \arrow[d, "\mathbf{W}_1"'] & \mathbf{z}_2 \arrow[r, "\mathbf{A}_3"] \arrow[d, "\mathbf{W}_2"'] & \cdots \arrow[r, "\mathbf{A}_T"] & \mathbf{z}_T \arrow[d, "\mathbf{W}_T"'] \\ \mathbf{x}_1 & \mathbf{x}_2 & & \mathbf{x}_T \end{tikzcd}$$

Figure 2: Time-varying LDS model

2.2.1 EM Algorithm

The latent states and parameters of LDS can be fit using the EM algorithm, the steps of which are outlined in *Pattern Recognition and Machine Learning 13.3* [1]. However, this formulation is for the time-invariant case (fixed $\mathbf{A}$ and $\mathbf{W}$) and for fitting only one sequence observations, whereas I have N. To account for these changes, I've re-derived the equations below. I can provide my full (very lengthy) derivation upon request.

E-step
In the E-step we run the inference algorithm to determine the posterior distribution of the latent variables $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{old})$. This step is composed of two parts: Kalman filtering and RTS smoothing.

1. **Kalman Filtering**

   Prediction Step: We get the predicted distribution: $P(\mathbf{z}_{n,t}|\mathbf{x}_{n,1:t-1}) = \mathcal{N}(\mathbf{z}_{n,t}|\boldsymbol{\mu}_{n,t}^{1:t-1}, \mathbf{V}_t^{1:t-1})$ by solving $\boldsymbol{\mu}_{n,t}^{1:t-1} = \mathbf{A}_t \boldsymbol{\mu}_{n,t-1}^{1:t-1}$ and $\mathbf{V}_t^{1:t-1} = \mathbf{A}_t \mathbf{V}_{t-1}^{1:t-1} (\mathbf{A}_t)^\top + \boldsymbol{\Sigma}^z$.

Measurement Step: We get the filtered distribution: $p(\mathbf{z}_{n,t}|\mathbf{x}_{n,1:t}) = \mathcal{N}(\mathbf{z}_{n,t}|\boldsymbol{\mu}_{n,t}^{1:t}, \mathbf{V}_t^{1:t})$ by solving $\boldsymbol{\mu}_{n,t}^{1:t} = \boldsymbol{\mu}_{n,t}^{1:t-1} + \mathbf{K}_t(\mathbf{r}_{n,t})$ and $\mathbf{V}_t^{1:t} = \mathbf{V}_t^{1:t-1} - \mathbf{K}_t\mathbf{W}_t\mathbf{V}_t^{1:t-1}$.
Where $\mathbf{r}_{n,t} = \mathbf{x}_{n,t} - \hat{\mathbf{x}}_{n,t}$ and $\hat{\mathbf{x}}_{n,t} = \mathbf{W}_t\boldsymbol{\mu}_{n,t}^{1:t-1}$. And the the Kalman Gain matrix $\mathbf{K}_t = \mathbf{V}_t^{1:t-1}(\mathbf{W}_t)^\top(\mathbf{S}_t)^{-1}$ where $\mathbf{S}_t = \mathbf{W}_t\mathbf{V}_t^{1:t-1}(\mathbf{W}_t)^\top + \boldsymbol{\Sigma}^x$.
Or equivalently: $\mathbf{K}_t = \mathbf{V}_t^{1:t}(\mathbf{W}_t)^\top(\mathbf{W}_t\mathbf{V}_t^{1:t}(\mathbf{W}_t)^\top + \boldsymbol{\Sigma}^x)^{-1}$.

2. **RTS Smoothing**
   We get the smoothed posterior distribution: $p(\mathbf{z}_{n,t}|\mathbf{x}_{n,1:T}) = \mathcal{N}(\mathbf{z}_{n,t}|\boldsymbol{\mu}_{n,t}^{1:T}, \mathbf{V}_t^{1:T})$ by solving $\boldsymbol{\mu}_{n,t}^{1:T} = \boldsymbol{\mu}_{n,t}^{1:t} + \mathbf{J}_t(\boldsymbol{\mu}_{n,t+1}^{1:T} - \boldsymbol{\mu}_{n,t+1}^{1:t})$ and $\mathbf{V}_t^{1:T} = \mathbf{V}_t^{1:t} + \mathbf{J}_t(\mathbf{V}_{t+1}^{1:T} - \mathbf{V}_{t+1}^{1:t})(\mathbf{J}_t)^\top$.
   Where the backward Kalman gain matrix is defined as $\mathbf{J}_t = \mathbf{V}_t^{1:t}(\mathbf{A}_{t+1})^\top(\mathbf{V}_{t+1}^{1:t})^{-1}$.

M-step
Now that we can estimate $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{old})$, we optimize $\boldsymbol{\theta}^{new}$ by maximizing $\mathcal{Q}$:

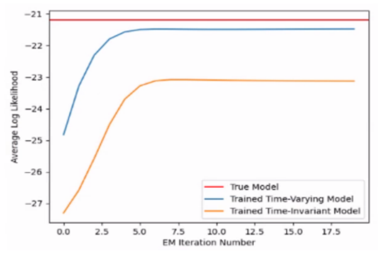
$$Q(\boldsymbol{\theta}^{new}, \boldsymbol{\theta}) = \mathbb{E}_{\mathcal{Z}|\mathcal{X},\theta}[\ln p(\mathcal{X}, \mathcal{Z}|\boldsymbol{\theta})] = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{\mathbf{Z}_n|\mathbf{X}_n,\boldsymbol{\theta}}[\ln p(\mathbf{X}_n, \mathbf{Z}_n|\boldsymbol{\theta})] \tag{4}$$

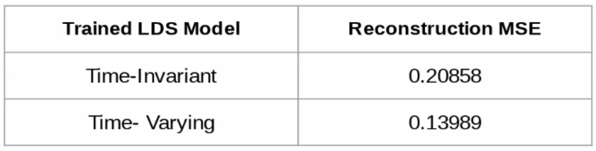
Where the log likelihood function for a single sample n is:

$$\ln p(\mathbf{X}_n, \mathbf{Z}_n|\theta) = \ln p(\mathbf{z}_{n,1}|\boldsymbol{\mu}_0, \mathbf{V}_0) + \sum_{t=2}^{T}\ln p(\mathbf{z}_{n,t}|\mathbf{z}_{n,t-1}, \mathbf{A}_t, \boldsymbol{\Sigma}^z) + \sum_{t=1}^{T}\ln p(\mathbf{x}_{n,t}|\mathbf{z}_{n,t}, \mathbf{W}_t, \boldsymbol{\Sigma}^x) \tag{5}$$

2.2.2 Implementation

I have implemented a flexible implementation of LDS in Python based on the PyKalman library [4]. I made two major updates to this framework, the first is I adapted it to be able to train on batches of sequences rather than just a single sequence. The second is I added a time varying version of LDS as only the time-invariant model was implemented. In the M-step, parameters are explicitly updated by taking the partial derivative of Q (Eq. 4) with respect to each parameter in $\boldsymbol{\theta}$, setting to 0 and solving.





Trained LDS Model	Reconstruction MSE
Time-Invariant	0.20858
Time- Varying	0.13989

Figure 3: Proof of effective EM implementation

I verified my EM implementation was correct using a simple test. First I initialized a true LDS model with known parameters that were time-dependent. I then sampled observations from this model to create a training set of 100 observations and testing set of 100 observations. Next I initialized a time-invariant and time-varying LDS model with default parameters and ran 20 EM iterations on each with the training observations. After each iteration, I computed the log likelihood of the test observations and compared it to that of the true model, the plot for which is in Figure 3. We can see as expected the time-varying model gets closer to the true model log likelihood. I also computed the reconstruction error as the mean square error (MSE) between the test observations and those reconstructed from the state/latent space by trained models.

3 Experiment

I generated an ellipse dataset that simulates sequences of 2D correspondence points. The shapes in the dataset have a x-radius that varies across observations but is the same across time and the y-radius varies cyclically across time but is consistent across observations. The LDS model should capture both variations. I created 100 training sequences of ellipses with 8 correspondence points at each time point where $T = 24$ (3 full cycles). The x-diameter is set by sampling the Gaussian distribution $\mathcal{N}(0.6, 0.1)$. The y-diameter follows the sine function: $0.4\sin(\frac{\pi}{4})t) + 0.6$. The period of this function is 8, so there a are 5 unique values the y-radius can take. I also added random noise with from Gaussian distribution $\mathcal{N}(0, 0.02)$ to the x and y point coordinates. Some partial examples are shown in Figure 4.

While the dimension of a time-point observation is $P = 16$ (eight 2D points), the intrinsic dimension is just two (x-radius and y-radius), thus I set the state/latent dimension to $L = 2$. I used a time-varying LDS model trained with 100 EM iterations. The log likelihood of the training observations over iterations can be seen in figure 4.

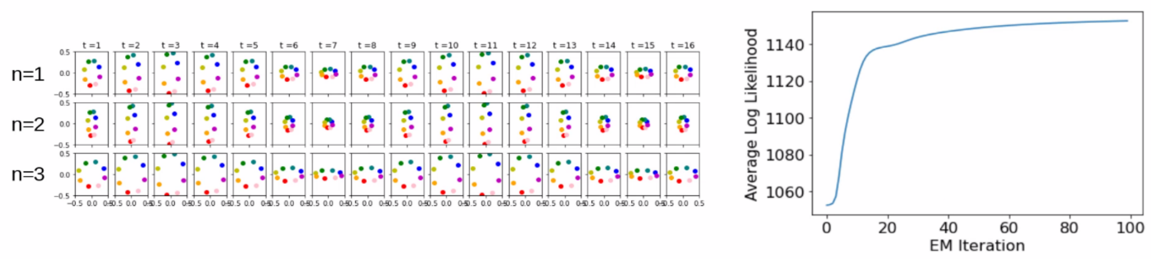


Figure 4: Left: Three examples of generated sequences where color is used to denote correspondence. Here we can see for each observation the x-radius is stationary while the y-radius changes sinusoidally. Right: Log likelihood of observations over EM iterations for fitting time-varying LDS.

4 Results

As a test set for the trained LDS model, I generated 100 new unseen ellipse correspondence point sequences. The accuracy of the model is analyzed using reconstruction, sampling, and inference means square error (MSE). **A note on scale: all ellipses have max diameter of 1, thus the largest possible mean square error that could occur is 1. So MSE values can be considered equivalent to relative MSE in analysis.**

4.1 Generalization

Ideally the time series model should be able to generalize from the training examples used in EM. It should be able to describe any valid instance of the class of shape, we can verify this on the test set. As done on the Gaussian data in Section 2.2.2, the reconstruction error can be calculated as the MSE between the test observations and those reconstructed from the state/latent space by the trained model. This is done by finding the state values (using the E-step) then applying Equation 2 to get the reconstructed observations. **The overall sequence reconstruction MSE was 0.000213.** As can be seen in figure 5, we also estimate the radius values from the true and reconstructed data and verify that they correlate. We can see the reconstruction error is very small and correlation is strong, thus the model generalizes well.

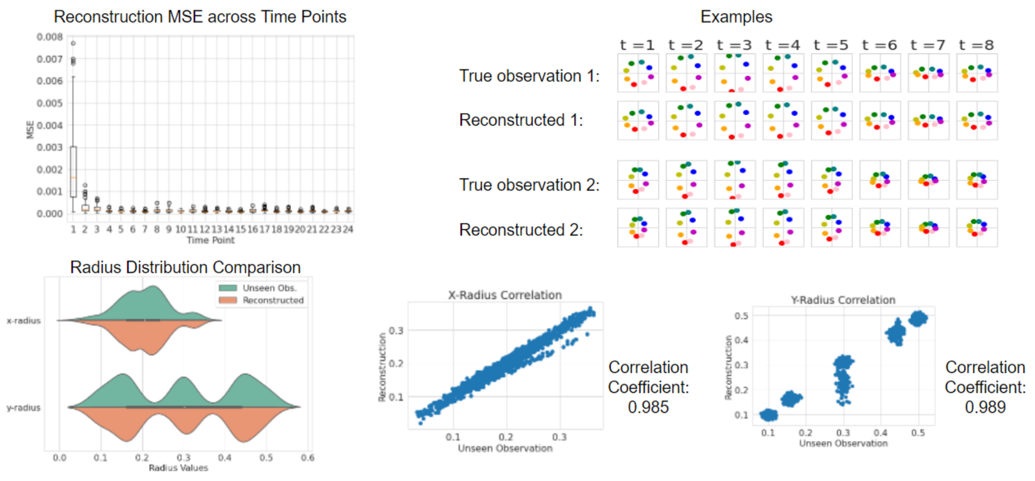


Figure 5: Generalization results show that LDS is very effective in reconstructing test sequences.

4.2 Specificity

Ideally the model should be specific to the training set, meaning it only represents valid instances of the sequences. This is where knowing the parameters of our observations comes in handy, because we can directly check if samples from LDS fit the requirements. To test this, we sample 100 new observations from the LDS model (examples of which can be seen in Figure 6). We then compute the MSE between these samples and the closest possible sequence that belongs to our true data distribution. That closest true sequence has the average x-values across the sampled sequence (because x-radius is time invariant) and the expected y-values given the sine function. The MSE across time-points is shown in Figure 6 and **the overall sample MSE was 0.002139**. Additionally, we can compute the x-radius and y-radius of the sampled observations and see if their distribution matches the distribution of radii from the training observations. In Figure 6, we can see that while the distribution of x-radius matches reasonably well, the distribution of y-radius does not. It appears LDS is unable to capture the multi-modal y-radius distribution, this limitation is caused by the Gaussian assumption.

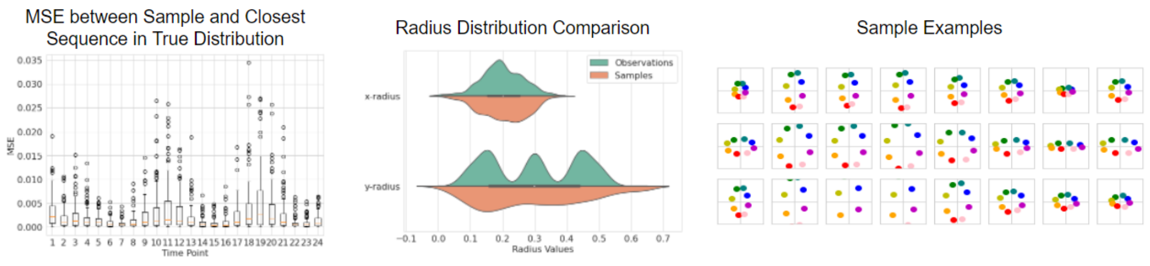


Figure 6: Specificity results demonstrate the degree to which samples match the expected data distribution.

4.3 Inference

The LDS model should also be able to accurately impute or infer missing time points, this would enable shape modeling on partial or incomplete sequences. To test this we randomly mask different percents of the time points in the test set and compute the MSE between the predicted and true values. To get the predicted values we simple fit the $\mathbf{z}$-values using know time-points in the E-step, then use Equation 2 to infer the missing time points. The results in Figure 7 show that LDS is able to accurately infer the missing time points. As expected the fewer missing time points, the better the accuracy. Figure 7 also shows some partial examples where the missing values are grayed out with their true value overlayed. Qualitatively we can see that the inferred values match the true values accurately.

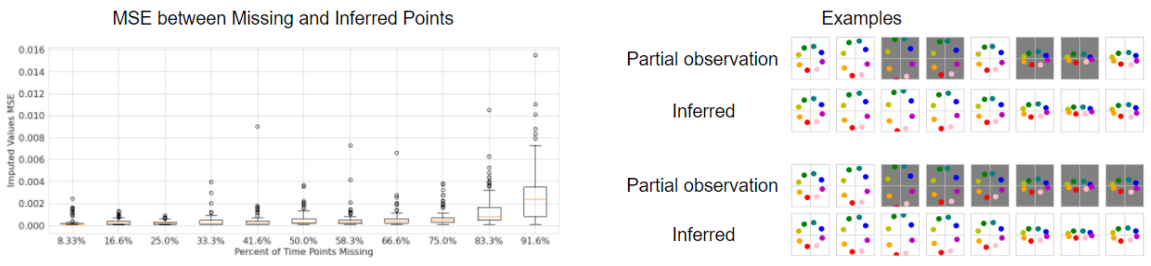


Figure 7: Inference results: LDS is able to correctly infer missing time points.

5 Conclusion

In conclusion, time-varying LDS trained via EM was able to accurately capture sequences of correspondence points. Though the specificity results showed us the limitations of the Gaussian assumption of LDS, the probabilistic framework has the benefit of being able to infer missing time points. The next steps for this work would be to derive the entropy of the complete data with respect to LDS so that the ShapeWorks optimization equation could be updated appropriately. If successful, this would enable spatiotemporal SSM, aiding in time-dependent pathology and disease detection. In future work, I could also look into other time-series prediction models, such as LSTM reccurrent neural networks, which would remove the restrictive assumptions of LDS potentially providing better predictions.

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] J. Cates, P Thomas Fletcher, M. Styner, M. Shenton, and R. Whitaker. Shape modeling and analysis with entropy-based particle systems. In *IPMI*, pages 333–345. Springer, 2007.

[3] Joshua Cates, Shireen Elhabian, and Ross Whitaker. Shapeworks: Particle-based shape correspondence and visualization software. In *Statistical Shape and Deformation Analysis*, pages 257–298. Elsevier, 2017.

[4] Daniel Duckworth. Pykalman: Inference for linear-gaussian systems. `https://github.com/pykalman/pykalman`, 2012.

[5] Nazli Sarkalkan, Harrie Weinans, and Amir A Zadpoor. Statistical shape and appearance models of bones. *Bone*, 60:129–140, 2014.

[6] 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(4):253–264, 1982.