EM Algorithms for Learning Local Dynamics Models in a Bayesian Framework

Samuel Otto

Outline

1. Motivation
2. Problem Setup
3. Maximum Likelihood and the EM Algorithm
4. Explicit Solutions for Maximization Step
   i. Weighted Regression
   ii. Weighted Gaussian Mixture
   iii. Categorical “Multinoulli” Distribution
5. Reconstructing the Estimates
6. Testing on a Chaotic Poincare Map
7. K-Means and Generalized K-Means as efficient EM variants
   i. Test on Chaotic Poincare Map
8. Extending to Nonlinear Local Dynamics using Kernels
   i. Test Variants on Chaotic Poincare Map
9. Extending to Non-Gaussian Density Estimation using Kernels
10. Future Work

Motivation

- Model complex, possibly chaotic dynamics using data.
  - Effect of control perturbations
  - Short-term prediction
  - Generative modeling: plausible dynamics sharing statistical properties
- Analyze dynamics locally and globally
  - Local spectral analysis
  - Identify global structures like orbits, tori, and attractors
- Understand the distribution of data points in phase space
  - Structure of data distribution in phase space
  - Identify regions of interest

Three Main Approaches

- If dynamics vary nonlinearly over phase space, we have three main approaches for modeling:

1. Use a single model with many nonlinear terms giving it enough representational capacity to capture dynamics everywhere.
   - EDMD: Use large number of nonlinear observables
   - Kernel Method: Kernel matrix grows with the square of the number of data points
2. Piece together many local models with fewer nonlinear terms in each
   - Models with fewer parameters can be updated and evaluated quickly
   - Representational power is increased simply by adding more models
3. Fully non-parametric representation using locally-weighted regression (Not practical)

Local Modeling Approach

- Model distribution of data in phase space using mixture of Gaussians
  - Universal approximator for continuous distributions
- To each Gaussian, we associate a model of the dynamics in that neighborhood
- Introduce a latent variable indicating which model describes the dynamics at a given point.
- Allows for new points to be classified according to maximum likelihood among the models
  - Finds the model most likely to capture dynamics at the new point
- Must learn the mixture of Gaussians and corresponding local dynamics together
  - Data clustering alone does not reflect the number of local models needed to capture dynamics in a region

Problem Setup

- Suppose we wish to approximate $y = f(x,u), x,y \in \mathbb{R}^n, u \in \mathbb{R}^q$ with training data $\{(x_j, u_j, y_j)\}_{j=1}^m$
- Introduce N models $\boldsymbol{M} = \{M^i = (f^i, {\mu_x}^i, {\mu_y}^i, R^i, \Sigma^i, \phi^i)\}_{i=1}^N$ of the form

$$Y^i = f^i(x,u) + V^i, \qquad V^i \sim p_{V^i}(v^i) = \mathcal{N}(v^i, \underline{0}, R^i)$$

- Introduce latent random variable $Z \in \{1, \dots, N\}$ indicating the model. The prior probabilities are assumed to be "Multinoulli"

$$P(Z = i) = \phi^i$$

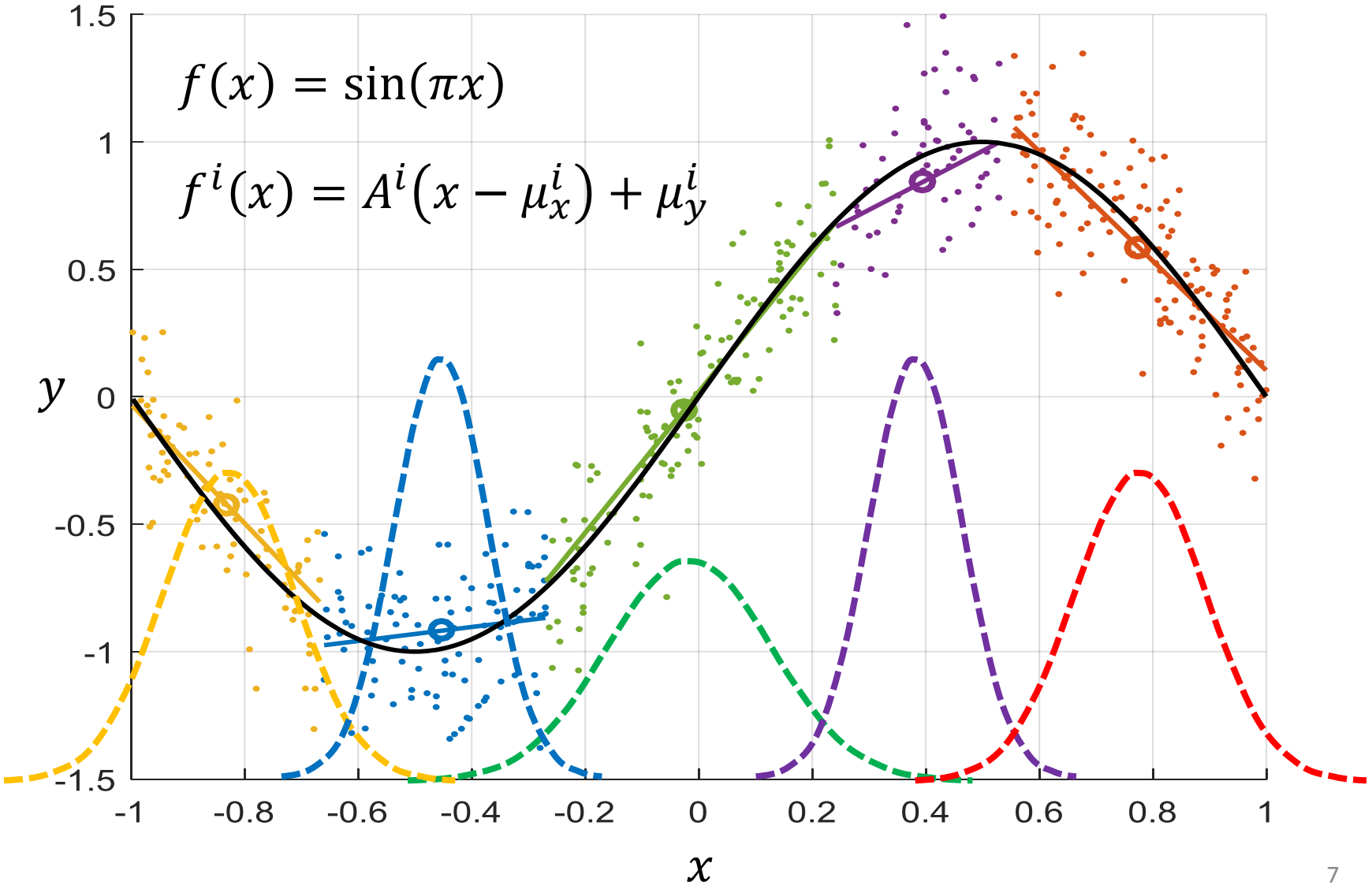
- Define posterior probabilities of the Gaussian mixture:

$$p_{X|Z=i}(x|Z = i) = \mathcal{N}(x, {\mu_x}^i, \Sigma^i)$$

- Infer conditional model probabilities using Bayes' Rule

$$P(Z = i|X = x) = \frac{p_{X|Z=i}(x|Z = i)P(Z = i)}{\sum_{k=1}^N p_{X|Z=k}(x|Z = k)P(Z = k)}$$

Problem Setup



Maximum Likelihood Estimation

- Likelihood function based on joint probability (m is number of training points)

$$L(\boldsymbol{M}) = \prod_{j=1}^{m} P_{model}(y_j, x_j; \boldsymbol{M})$$

- Training on log likelihood is equivalent to minimizing cross entropy

$$l(\boldsymbol{M}) = \mathbb{E}_{X,Y\sim\hat{P}_{data}}[\log P_{model}(X, Y; \boldsymbol{M})] = \frac{1}{m}\sum_{j=1}^{m} \log P_{model}(y_j, x_j; \boldsymbol{M})$$

- Condition on latent variable Z and use total probability (N is number of models)

$$P_{model}(y_j, x_j; \boldsymbol{M}) = \sum_{i=1}^{N} P(y_j, x_j, Z = i)$$

- Chain rule for conditional probability

$$P_{model}(y_j, x_j; \boldsymbol{M}) = \sum_{i=1}^{N} p_{Y|Z,X,u}(y_j|u_j, x_j, Z = i)p_{X|Z=i}(x_j|Z = i)P(Z = i)$$

$$P_{model}(y_j, x_j; \boldsymbol{M}) = \sum_{i=1}^{N} p_{V^i}(y_j - f^i(x_j, u_j))p_{X|Z=i}(x_j|Z = i)P(Z = i)$$

Maximum Likelihood Estimation

- Introduce a yet to be determined distribution over the models at each training point $Q_j(Z)$

$$l(\boldsymbol{M}) = \frac{1}{m}\sum_{j=1}^{m} \log\left[\sum_{i=1}^{N} Q_j(Z=i)\frac{p_{V^i}(v_j^i)p_{X|Z=i}(x_j|Z=i)P(Z=i)}{Q_j(Z=i)}\right]$$

- Observe that the inner sum is an expectation

$$l(\boldsymbol{M}) = \frac{1}{m}\sum_{j=1}^{m} \log\left(\mathbb{E}_{Z\sim Q_j}\left[\frac{p_{V^Z}(v_j^Z)p_{X|Z}(x_j|Z)P(Z)}{Q_j(Z)}\right]\right)$$

- Use Jensen's inequality to lower bound the log likelihood

$$l(\boldsymbol{M}) \geq \tilde{l}(\boldsymbol{M}) = \frac{1}{m}\sum_{j=1}^{m} \mathbb{E}_{Z\sim Q_j}\left[\log\left(\frac{p_{V^Z}(v_j^Z)p_{X|Z}(x_j|Z)P(Z)}{Q_j(Z)}\right)\right]$$

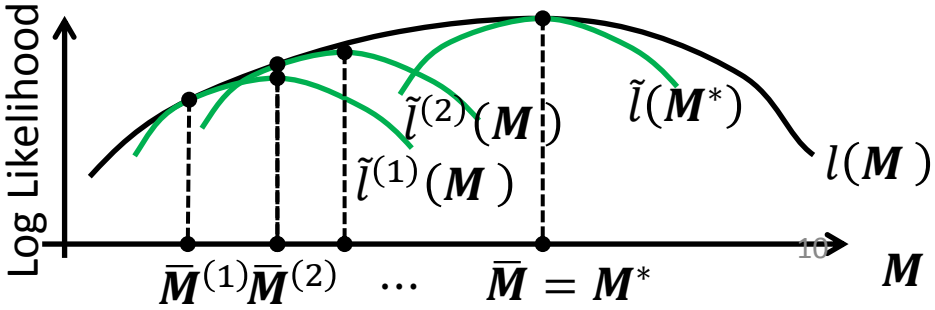
$$\boxed{l(\boldsymbol{M}) \geq \tilde{l}(\boldsymbol{M}) = \frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{N} Q_j(Z=i)\log\left(\frac{p_{V^i}(v_j^i)p_{X|Z=i}(x_j|Z=i)P(Z=i)}{Q_j(Z=i)}\right)}$$

- Important note: this inequality becomes an equality when the inside term does not depend on Z

The EM Algorithm

- Iterative update process for the model parameters $\boldsymbol{M}$ given lower bounds using previous set of parameters $\bar{\boldsymbol{M}}$.
  - Maximizing lower bounds $\tilde{l}(\boldsymbol{M})$ gives monotone increase in log likelihood function.
- Clever choice for $Q_j(Z)$ such that the inequality becomes equality when the algorithm has converged $\bar{\boldsymbol{M}} = \boldsymbol{M}^*$.

$$Q_j(Z = i; \bar{\boldsymbol{M}}) = \frac{p_{V^i}\left(v_j^i; \bar{\boldsymbol{M}}\right) p_{X|Z=i}\left(x_j \middle| Z = i; \bar{\boldsymbol{M}}\right) P(Z = i; \bar{\boldsymbol{M}})}{\sum_{k=1}^{N} p_{V^k}\left(v_j^k; \bar{\boldsymbol{M}}\right) p_{X|Z=k}\left(x_j \middle| Z = k; \bar{\boldsymbol{M}}\right) P(Z = k; \bar{\boldsymbol{M}})}$$



The EM Algorithm

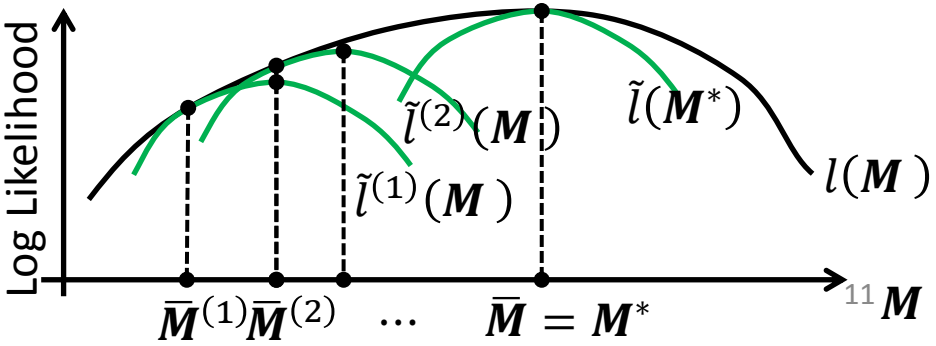
1. Expectation Step: Given previous set of parameter estimates $\bar{\boldsymbol{M}}$, compute the probabilities $Q_j(Z = i; \bar{\boldsymbol{M}})$ to use in the expectation

$$\bar{Q}_j^i \triangleq Q_j(Z = i; \bar{\boldsymbol{M}}) = \frac{p_{V^i}\left(v_j^i; \bar{\boldsymbol{M}}\right) p_{X|Z=i}\left(x_j \middle| Z = i; \bar{\boldsymbol{M}}\right) P(Z = i; \bar{\boldsymbol{M}})}{\sum_{k=1}^{N} p_{V^k}\left(v_j^k; \bar{\boldsymbol{M}}\right) p_{X|Z=k}\left(x_j \middle| Z = k; \bar{\boldsymbol{M}}\right) P(Z = k; \bar{\boldsymbol{M}})}$$

2. Maximization Step: Maximize the lower bound $\tilde{l}(\boldsymbol{M})$ using the above probabilities in the expectation

$$\bar{\boldsymbol{M}} \leftarrow \underset{\boldsymbol{M}}{\operatorname{argmax}} \frac{1}{m} \sum_{j=1}^{m} \sum_{i=1}^{N} \bar{Q}_j^i \log\left(\frac{p_{V^i}\left(v_j^i; \boldsymbol{M}\right) p_{X|Z=i}\left(x_j \middle| Z = i; \boldsymbol{M}\right) P(Z = i; \boldsymbol{M})}{\bar{Q}_j^i}\right)$$

3. Iterate until convergence



Maximization Step

- Helpful because we now have the log of a product which decouples into separate optimizations. Substituting the known forms we have

$$\bar{\boldsymbol{M}} \leftarrow \underset{\boldsymbol{M}}{\operatorname{argmax}} \underbrace{\sum_{j=1}^{m}\sum_{i=1}^{N} \bar{Q}_j^i \log\left[\mathcal{N}\left(y_j, f^i(x_j,u_j), R^i\right)\right]}_{(\boldsymbol{I})} + \underbrace{\sum_{j=1}^{m}\sum_{i=1}^{N} \bar{Q}_j^i \log\left[\mathcal{N}\left(x_j, \mu_x^i, \Sigma^i\right)\right]}_{(II)} + \underbrace{\sum_{j=1}^{m}\sum_{i=1}^{N} \bar{Q}_j^i \log\left(\phi^i\right)}_{(III)}$$

I. Weighted regression

$$\underset{\boldsymbol{M}}{\operatorname{argmax}}(I) = \underset{\{(f^i,R^i)\}_{i=1}^N}{\operatorname{argmax}} \sum_{j=1}^{m}\sum_{i=1}^{N} \bar{Q}_j^i \left[-\frac{1}{2}\log\left|R^i\right| - \frac{1}{2}\left(y_j - f^i(x_j,u_j)\right)^T \left(R^i\right)^{-1}\left(y_j - f^i(x_j,u_j)\right)\right]$$

II. Weighted mixture of Gaussians

$$\underset{\boldsymbol{M}}{\operatorname{argmax}}(II) = \underset{\{(\mu_x^i,\Sigma^i)\}_{i=1}^N}{\operatorname{argmax}} \sum_{j=1}^{m}\sum_{i=1}^{N} \bar{Q}_j^i \left[-\frac{1}{2}\log\left|\Sigma^i\right| - \frac{1}{2}\left(x_j - \mu_x^i\right)^T \left(\Sigma^i\right)^{-1}\left(x_j - \mu_x^i\right)\right]$$

III. Maximum likelihood "Multinoulli"

$$\underset{\boldsymbol{M}}{\operatorname{argmax}}(III) = \underset{\{\phi^i\}_{i=1}^N}{\operatorname{argmax}} \sum_{j=1}^{m}\sum_{i=1}^{N} \bar{Q}_j^i \log\phi^i \quad s.t. \quad 1 = \sum_{i=1}^{N} \phi^i$$

(I) Weighted Regression

- Form vector with (possibly nonlinear) observables $\theta_1, \theta_2, \dots, \theta_L : \mathbb{R}^{n+q} \to \mathbb{R}$ of the state and control

$$\Phi_j \triangleq \begin{bmatrix} \theta_1(x_j, u_j) & \cdots & \theta_L(x_j, u_j) \end{bmatrix}^T \quad or \quad \Phi_j \triangleq \begin{bmatrix} x_j & u_j & 1 \end{bmatrix}^T$$

- The learned functions are then linear combinations of observables

$$f^i(x,u) = A^i \Phi(x,u) = A^i \begin{bmatrix} \theta_1(x_j, u_j) & \cdots & \theta_L(x_j, u_j) \end{bmatrix}^T$$

- Take derivatives and do math to optimize (I) analytically.

$$A^i = \left(\sum_{j=1}^m \bar{Q}_j^i y_j \Phi_j^T \right) \left(\sum_{j=1}^m \bar{Q}_j^i \Phi_j \Phi_j^T \right)^+ \qquad R^i = \frac{\sum_{j=1}^m \bar{Q}_j^i (y_j - A^i \Phi_j)(y_j - A^i \Phi_j)^T}{\sum_{j=1}^m \bar{Q}_j^i}$$

- Letting $\boldsymbol{Y} = [y_1 \cdots y_m]$, $\boldsymbol{\Phi} = [\Phi_1 \cdots \Phi_m]$, and $W^i = diag\left[\bar{Q}_1^i, \dots, \bar{Q}_m^i\right]$ we have the familiar weighted least squares equations

$$\boxed{A^i = \left(\boldsymbol{Y} W^i \boldsymbol{\Phi}^T\right)\left(\boldsymbol{\Phi} W^i \boldsymbol{\Phi}^T\right)^+ \quad R^i = \left(\mathbf{1}_m^T W^i \mathbf{1}_m\right)^{-1} \left(\boldsymbol{Y} - A^i \boldsymbol{\Phi}\right) W^i \left(\boldsymbol{Y} - A^i \boldsymbol{\Phi}\right)^T}$$

(II) Weighted Mixture of Gaussians

- Taking derivatives and optimizing (II) gives the weighted mixture of Gaussians model
- Let $\boldsymbol{X} = [x_1 \cdots x_m]$ and $W^i = diag[\bar{Q}_1^i, \ldots, \bar{Q}_m^i]$
- The centers are found:

$$\boxed{\mu_x^i = \frac{\sum_{j=1}^m \bar{Q}_j^i x_j}{\sum_{j=1}^m \bar{Q}_j^i} = \frac{\boldsymbol{X} W^i \mathbf{1}_m}{\mathbf{1}_m^T W^i \mathbf{1}_m}}$$

- The covariances are found:

$$\boxed{\Sigma^i = \frac{\sum_{j=1}^m \bar{Q}_j^i \left(x_j - \mu_x^i\right)\left(x_j - \mu_x^i\right)^T}{\sum_{j=1}^m \bar{Q}_j^i} = \frac{\left(\boldsymbol{X} - \mu_x^i \mathbf{1}_m^T\right) W^i \left(\boldsymbol{X} - \mu_x^i \mathbf{1}_m^T\right)^T}{\mathbf{1}_m^T W^i \mathbf{1}_m}}$$

(III) Weighted Multinoulli

- Form the Lagrangian to include the constraint

$$\mathcal{L}(\phi^1, \ldots, \phi^N, \lambda) = \sum_{j=1}^{m} \sum_{i=1}^{N} \bar{Q}_j^i \log \phi^i + \lambda \left(1 - \sum_{i=1}^{N} \phi^i \right)$$

- Setting derivatives to zero and doing some algebra gives the solution

$$\boxed{\phi^i = \frac{1}{m} \sum_{j=1}^{m} \bar{Q}_j^i}$$

- In the expressions for the other solutions, we have the normalization constant $\mathbf{1}_m^T W^i \mathbf{1}_m = \sum_{j=1}^{m} \bar{Q}_j^i = m\phi^i$ in the denominator
  - This is the expected number of training points belonging to each model a priori.

Reconstructing the Dynamics

- Given a new point x and control input u, how do we predict the dynamics?
- Conditional model probabilities using Bayes' rule

$$P(Z = i|X = x) = \frac{\mathcal{N}\left(x, \mu_x^i, \Sigma^i\right)\phi^i}{\sum_{k=1}^{N} \mathcal{N}\left(x, \mu_x^k, \Sigma^k\right)\phi^k}$$

1. Maximum likelihood model assignment

$$\hat{y} = f^{i^*}(x, u) \quad i^* = \underset{i}{\operatorname{argmax}}\, P(Z = i|X = x)$$

2. Stochastic model assignment

$$\hat{y} = f^{i^*}(x, u) \quad i^* \sim P(Z|X = x)$$

3. Weighted model assignment

$$\hat{y} = \mathbb{E}[Y] = \sum_{i=1}^{N} P(Z = i|X = x)\mathbb{E}\left[Y^i\right] = \sum_{i=1}^{N} P(Z = i|X = x) f^i(x, u)$$

4. Hybrid weighted model assignment from top $N' \leq N$ models
   - Sort models $(i_1, i_2, \ldots, i_N)$ from greatest to least probability $P(Z|X = x)$

$$\hat{y} = \left(\sum_{l=1}^{N'} P(Z = i_l|X = x)\right)^{-1} \sum_{l=1}^{N'} P(Z = i_l|X = x) f^{i_l}(x, u)$$

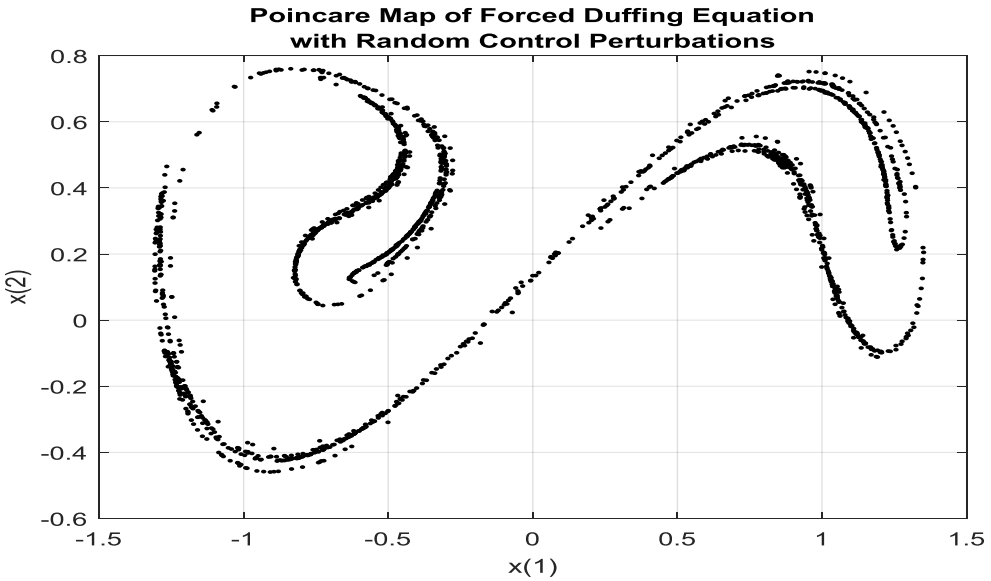
5. Some kind of Kriging may also be possible. This is a topic for future work.

Application to Duffing Equation with Control

- Forced Duffing equation with control input

$$\frac{d}{dt}\begin{bmatrix}x(1)\\x(2)\end{bmatrix}=\begin{bmatrix}x(2)\\\gamma\cos(\omega t)-\delta x(2)-\alpha x(1)-\beta x(1)^3\end{bmatrix}+\begin{bmatrix}0 & 0 & 0 & 0 & 0\\1 & \cos\omega t & \sin\omega t & \cos 2\omega t & \sin 2\omega t\end{bmatrix}\begin{bmatrix}u(1)\\u(2)\\u(3)\\u(4)\\u(5)\end{bmatrix}$$

- Define time $T=2\pi/\omega$ Poincare map with constant control u over each interval $x_{n+1}=f(x_n,u_n)$
- The Poincare map was simulated for $m=2000$ iterations with small uniformly distributed random control perturbations $u(k)\in[-10^{-3},10^{-3}]$



Error Metrics

- The algorithms will be evaluated on $m_{test} = 1000$ new testing points generated from new random control perturbations
- Mean estimation error distance

$$\bar{E}_D = \frac{1}{m_{test}} \sum_{j=1}^{m_{test}} \left\| y_j - \hat{y}_j \right\|$$

- Decaying error norm after many steps. Start at x_0 and iterate the map f and approximate map $\hat{f}$ with random control inputs $u_0, u_2, \ldots, u_{M-1}$ generating sequences $x_1, x_2, \ldots, x_M$ and $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_M$ respectively

$$E_\alpha^M(x_0) = \sum_{n=1}^{M} \frac{\|x_n - \hat{x}_n\|}{\alpha^{n-1}}$$

  - The average is taken over the data distribution by sampling many initial points

$$\bar{E}_\alpha^M = \frac{1}{m_{test}} \sum_{j=1}^{m_{test}} E_\alpha(x_j)$$

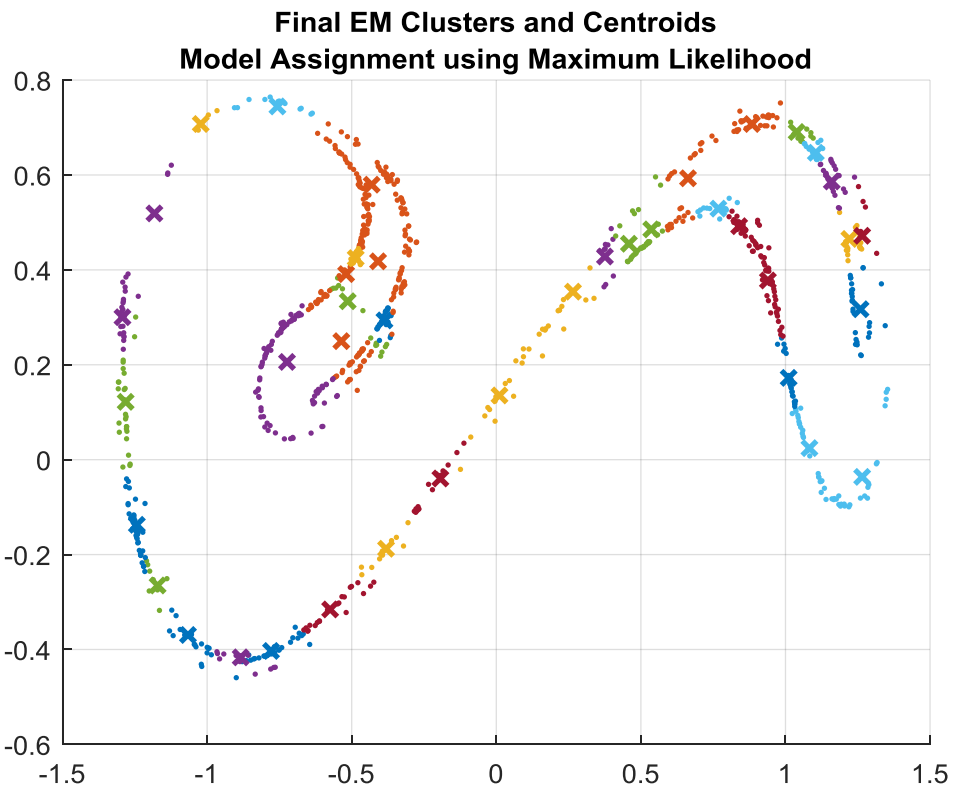
- The autocorrelations of the uncontrolled dynamics over many iterations will be compared in order to estimate stationary statistics $R(n) = \mathbb{E}\left[x_k x_{k-n}^T\right]$ of the actual and $\hat{R}(n) = \mathbb{E}\left[\hat{x}_k \hat{x}_{k-n}^T\right]$ of the modeled dynamics.
- The histograms of the distance errors on the 1000 test points will also be considered

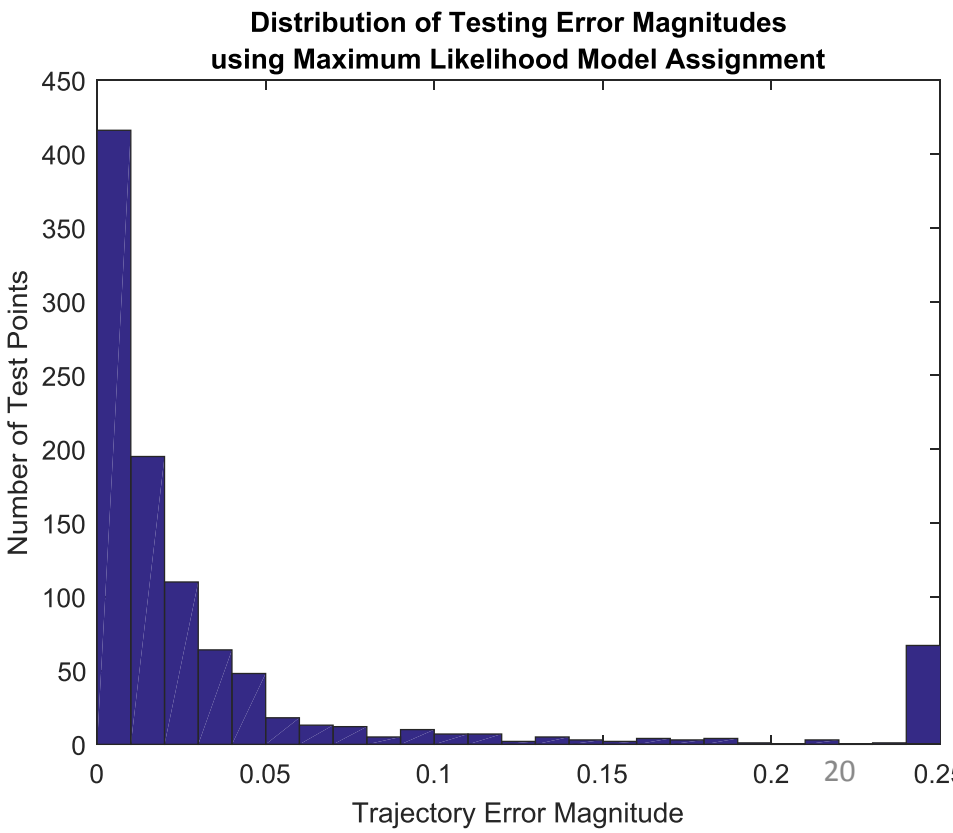
Linear EM Gaussian Mixture Results

- $N = 40$ linear models were fit
- Each model is fit to the $m' = 500$ training points with the largest weights $\bar{Q}_j^i$
- Maximum likelihood model assignment was used to reconstruct the dynamics
  - Numerical experiments show that of the methods discussed, maximum likelihood produced the lowest errors.
- Minimum singular values of covariances Σ and R were needed to prevent degeneracy
- $m_{test} = 1000$ independently generated test points with random control perturbations were used for evaluation

Linear EM Gaussian Mixture Results

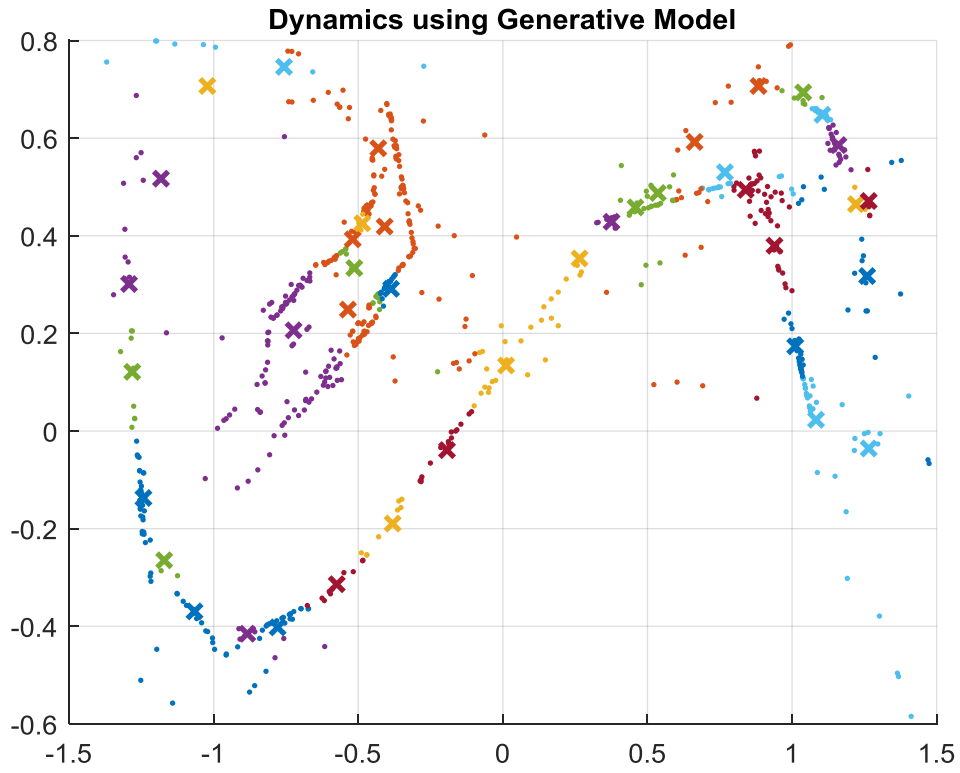
- Mean Distance Error: $\boxed{\bar{E}_D = 0.0608}$
- Median Distance Error: 0.0134
- Decaying Error Norm over $M = 10$ steps with $\alpha = 2$: $\boxed{\bar{E}_\alpha^M = 0.3921}$

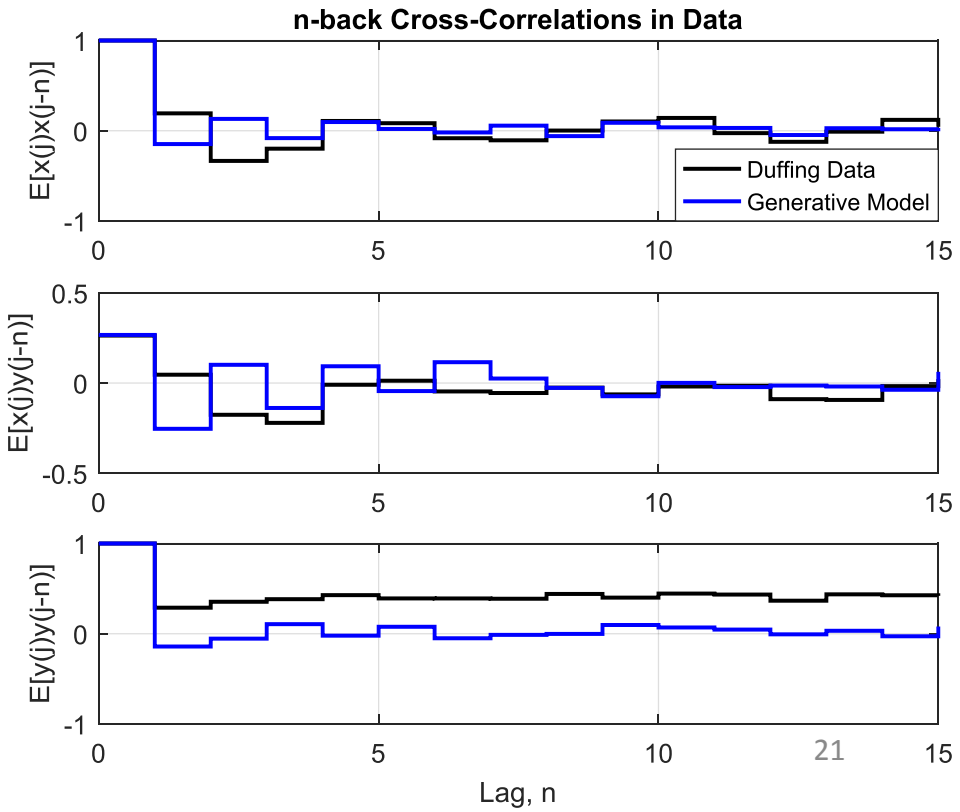




Use as a Generative Model

- The dynamics were simulated using the learned model for 1000 steps with no control perturbations
- The autocorrelation was compared to the actual Poincare map





Simplified EM: K-Means

- Rather than letting a training point x_j belong to several models, we force it to belong to only one
  - Choose the most likely model, $z_j = \text{argmax}_{i=1,\ldots,N} Q_j(Z = i; \bar{\boldsymbol{M}})$
  - Equivalent to using sparse $\bar{Q}_j^i = \begin{cases} 1 & i = z_j \\ 0 & i \neq z_j \end{cases}$
  - Speeds up regression and Gaussian mixture since fewer points are used to fit each model

1. Expectation Step: Find the most likely model at each training point

$$z_j \triangleq \underset{i=1,\ldots,N}{\text{argmax}}\, p_{V^i}\left(v_j^i; \bar{\boldsymbol{M}}\right) p_{X|Z=i}\left(x_j \middle| Z = i; \bar{\boldsymbol{M}}\right) P(Z = i; \bar{\boldsymbol{M}})$$

   - The collection of points assigned to each model will be used

$$S^i \triangleq \left\{j \in \{1,2,\ldots,m\} : z_j = i\right\}$$

2. Maximization Step: The usual maximization, but with sparse $\bar{Q}_j^i$

$$\bar{\boldsymbol{M}} \leftarrow \underset{\boldsymbol{M}}{\text{argmax}} \sum_{i=1}^{N} \sum_{j \in S^i} \log\left[p_{V^i}\left(v_j^i; \boldsymbol{M}\right) p_{X|Z=i}\left(x_j \middle| Z = i; \boldsymbol{M}\right) P(Z = i; \boldsymbol{M})\right]$$

K-Means Maximization Step

- Identical to EM maximization step, but with sparse weight matrices.

I. Regression: let $\boldsymbol{Y}^i = [\cdots \quad y_j \quad \cdots]_{j \in S^i}$ and $\boldsymbol{\Phi}^i = \begin{bmatrix} & \theta_1(x_j, u_j) & \\ \cdots & \vdots & \cdots \\ & \theta_L(x_j, u_j) & \end{bmatrix}_{j \in S^i}$

  - We have simple least squares over a restricted set of points

$$A^i = \left(\boldsymbol{Y}^i \boldsymbol{\Phi}^{i^T}\right)\left(\boldsymbol{\Phi}^i \boldsymbol{\Phi}^{i^T}\right)^+ \qquad R^i = \frac{1}{|S^i|}\left(\boldsymbol{Y}^i - A^i \boldsymbol{\Phi}^i\right)\left(\boldsymbol{Y}^i - A^i \boldsymbol{\Phi}^i\right)^T$$

II. Gaussian Mixture:

  - We fit the maximum likelihood Gaussian to each point cluster

$$\mu_x^i = \frac{1}{|S^i|}\sum_{j \in S^i} x_j \qquad \Sigma^i = \frac{1}{|S^i|}\sum_{j \in S^i}\left(x_j - \mu_x^i\right)\left(x_j - \mu_x^i\right)^T$$

III. Multinoulli:

  - Prior model probabilities are just the fraction of points in each cluster

$$\phi^i = \frac{1}{m}\left|S^i\right|$$

K-Means/EM Variants

Many variants are possible. For example:

1. Choose most likely model $z_j = \operatorname{argmax}_{i=1,\ldots,N} Q_j(Z = i; \bar{\boldsymbol{M}})$ but keep the weights $\bar{Q}_j^i = \begin{cases} Q_j(Z = i; \bar{\boldsymbol{M}}) & i = z_j \\ 0 & i \neq z_j \end{cases}$ ➔ Weighted regression and Gaussian mixture.
2. Choose the top $N' < N$ models based on likelihood $Q_j(Z = i; \bar{\boldsymbol{M}})$ and zero-out the rest (call it "Generalized K-Means")
3. Allocate training points by choosing among top $N' < N$ models based on minimum prediction error $z_j = \operatorname{argmin}_{i=i_1,\ldots,i_{N'}} \|y_j - f^i(x_j, u_j)\|$
4. For each model (i) restrict to top $m' < m$ training points S^i and keep the weights on these points only (call it "Restricted EM")

$$\bar{Q}_j^i = \begin{cases} Q_j(Z = i; \bar{\boldsymbol{M}}) & j \in S^i \\ 0 & j \notin S^i \end{cases}$$

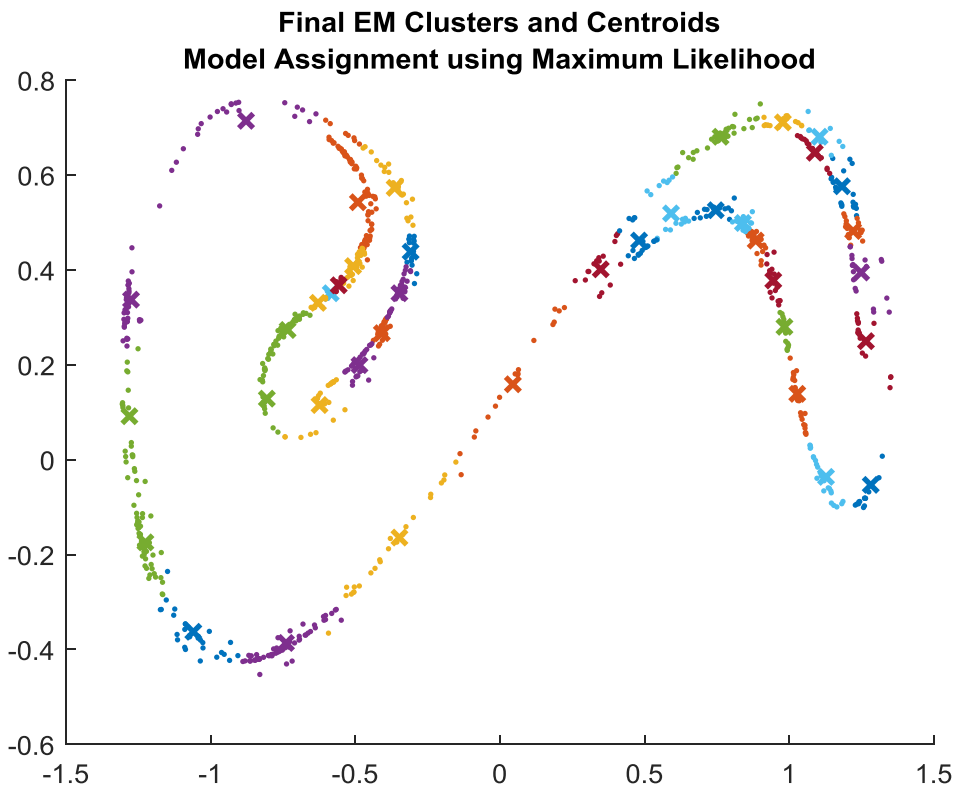
- These will enable efficient extensions to kernel regression.
- Intuitive since *our models only need to be accurate locally*.

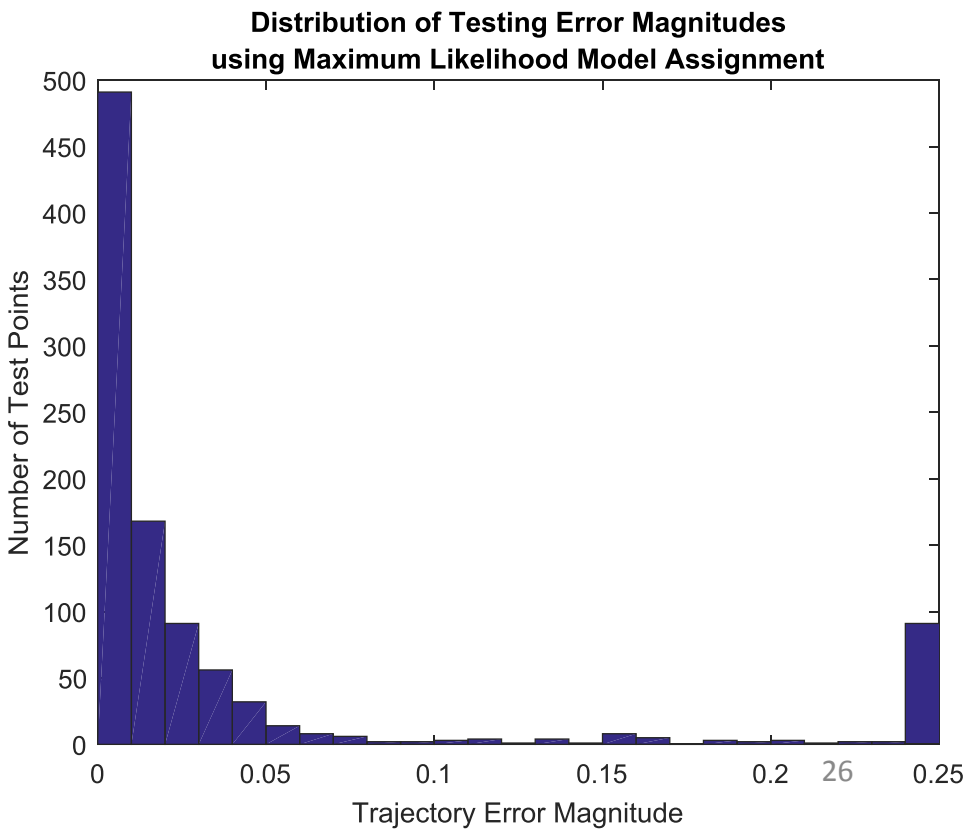
Linear K-Means Gaussian Mixture Results

- $N = 40$ linear models were fit
- Each point is assigned to the single most likely model at each step
- Maximum likelihood model assignment is used to reconstruct the dynamics
  - Numerical experiments show that of the methods discussed, maximum likelihood produced the lowest errors.
- Minimum singular values of covariances Σ and R were needed to prevent degeneracy
- $m_{test} = 1000$ independently generated test points with random control perturbations were used for evaluation

Linear K-Means Gaussian Mixture Results

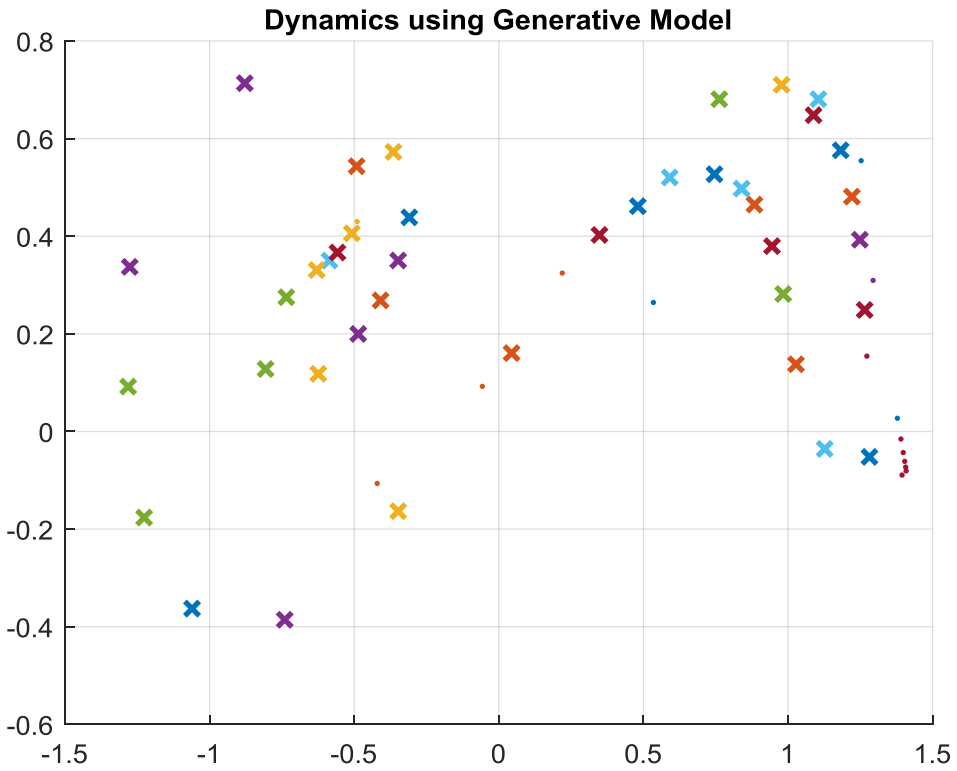
- Mean Distance Error: $\boxed{\bar{E}_D = 0.0842}$
- Median Distance Error: 0.0103
- Decaying Error Norm over $M = 10$ steps with $\alpha = 2$: $\boxed{\bar{E}_\alpha^M = 342.6198}$





Use as a Generative Model

- The dynamics were simulated using the learned model for 1000 steps with no control perturbations
- The autocorrelation was compared to the actual Poincare map
- The approximation diverges and is therefore not useful as a generative model

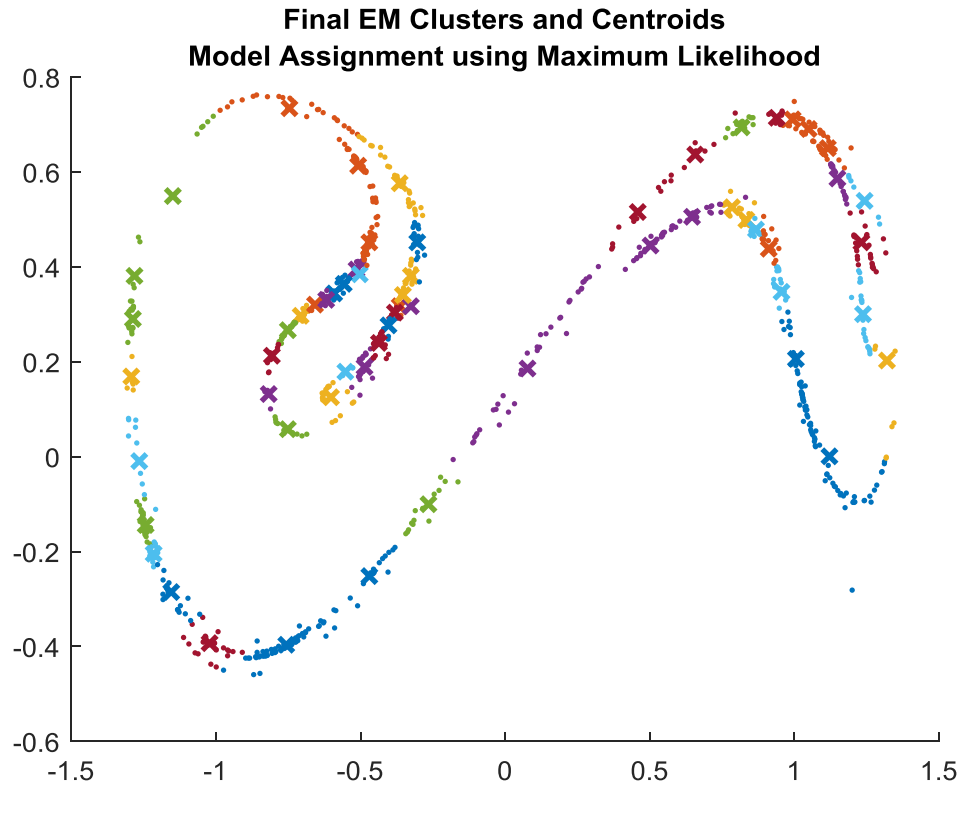


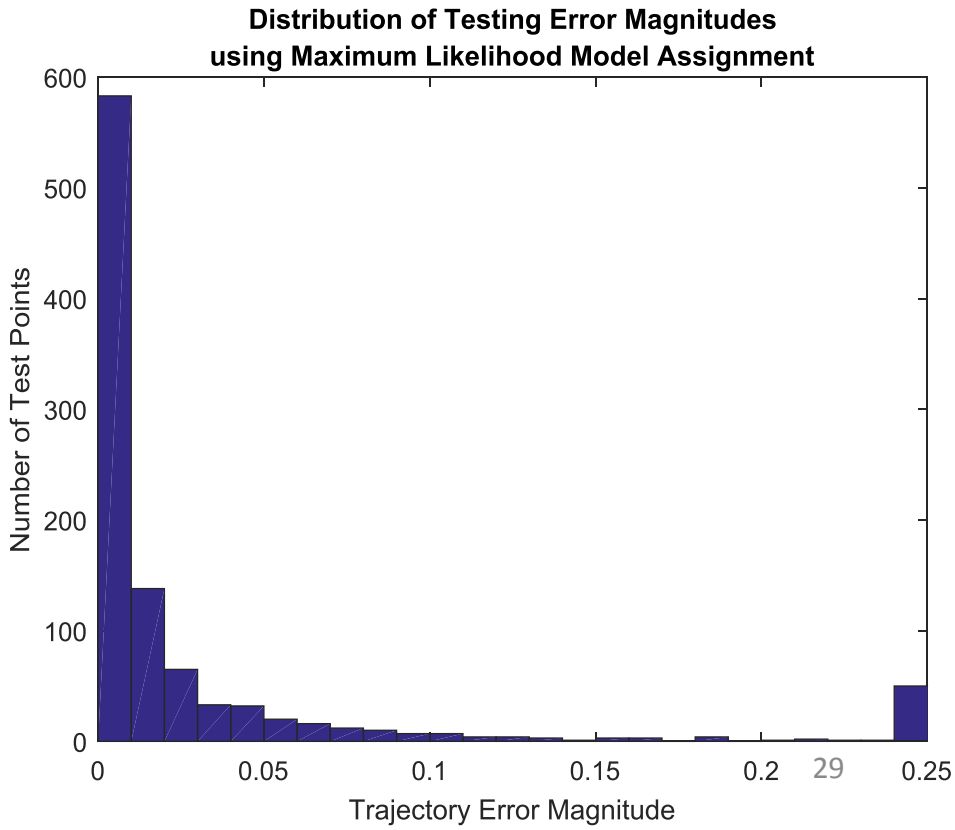
Linear Generalized K-Means Results

- $N = 60$ linear models were fit
- Each training point was assigned to $N' < N$ models based on maximum likelihood giving sparse $\bar{Q}_j^i$
- K-means was initially converged with $N' = 1$ then increased to $N' = 3$ and converged
  - This encourages the centroids to stay tight to the filamented distribution of points in the Poincare map
- Maximum likelihood model assignment is used to reconstruct the dynamics
- Minimum singular values of covariances Σ and R were needed to prevent degeneracy

Linear Generalized K-Means Results

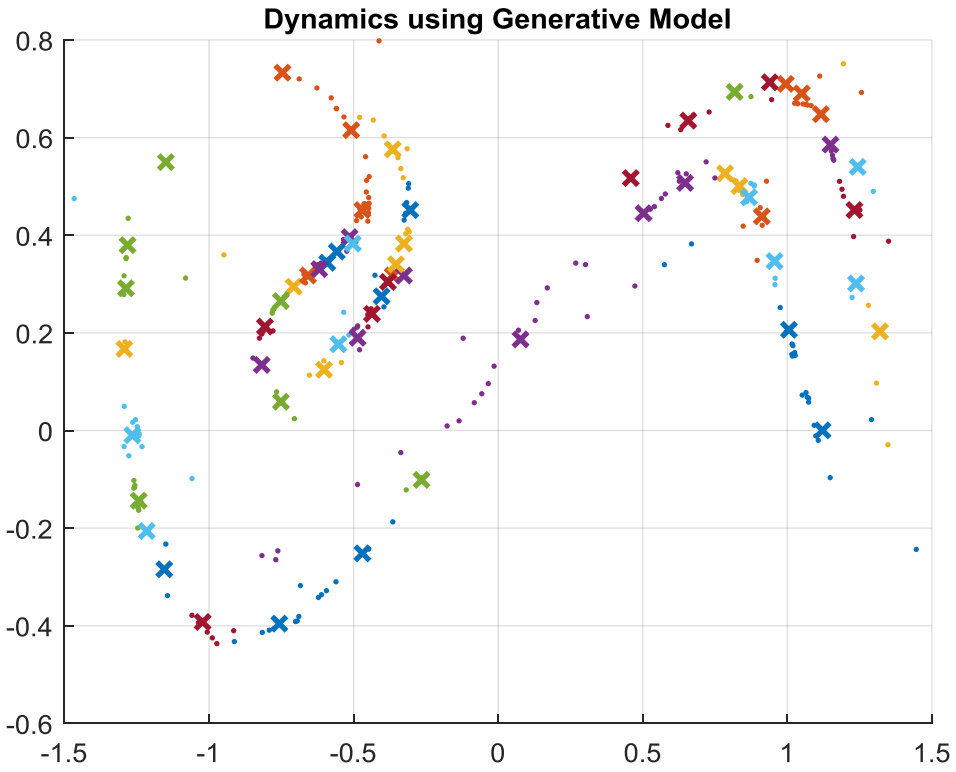
- Mean Distance Error: $\boxed{\bar{E}_D = 0.0498}$
- Median Distance Error: 0.0068
- Decaying Error Norm over $M = 10$ steps with $\alpha = 2$: $\boxed{\bar{E}_\alpha^M = 0.5541}$

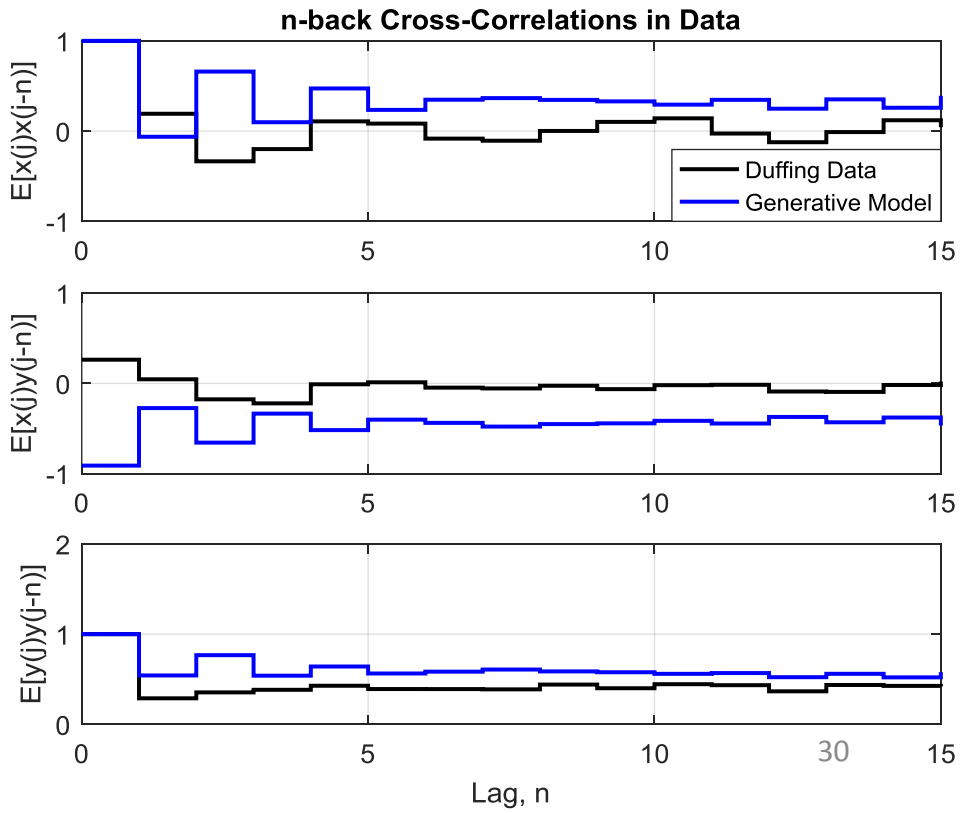




Use as a Generative Model

- The dynamics were simulated using the learned model for 1000 steps with no control perturbations
- The autocorrelation was compared to the actual Poincare map





Highlights from Linear Cases

- Generalized K-means where points can be assigned to multiple models attains the lowest error with a large number of models
- The K-means and generalized K-means centroids fit the data more tightly than EM.
- K-means and generalized K-means both take less time to converge than EM even with the restricted set of points used to train each model.
- Linear models are very useful
  - Local DMD
- Linear models are still too inaccurate to replicate the dynamics over multiple steps
  - Nonlinear kernel models will now be developed

Extending to Nonlinear Local Dynamics

- As we have seen, the local dynamics can be highly nonlinear
  - The number of linear models that can be fit is limited by the number of training points
  - Cannot provide adequate resolution in areas with high nonlinearity unless they are accompanied by a high density of training point
- We would like to model nonlinear local variations efficiently without over-fitting.
- Must be able to include local weighting when fitting the nonlinear model
- These requirements suggest using *weighted kernel ridge regression*.

Weighted Kernel Ridge Regression

- Our solution to the regression problem posed probabilistically

$$\hat{y} = f^i(x,u) = \left(\boldsymbol{Y}W^i\boldsymbol{\Phi}^T\right)\left(\boldsymbol{\Phi}W^i\boldsymbol{\Phi}^T\right)^+\Phi(x,u)$$

Is the solution of an equivalent matrix optimization problem

$$\hat{y} = A^i\Phi(x,u) \quad A^i = \underset{A\in\mathbb{R}^{n\times L}}{\operatorname{argmin}} \left\|(\boldsymbol{Y} - A\boldsymbol{\Phi})\sqrt{W^i}\right\|_F^2$$

- We introduce nonlinearity by considering a nonlinear feature map from the data $\xi = [x \quad u \quad 1]^T$ into a high-dimensional Hilbert space $\mathcal{H}$

$$\Phi\colon \mathbb{R}^{n+q+1} \to \mathcal{H} \qquad \boldsymbol{\Phi}\colon \mathbb{R}^m \to \mathcal{H}$$

$$\xi \mapsto \Phi(\xi) \qquad v \mapsto v_1\Phi(\xi_1) + \cdots + v_m\Phi(\xi_m)$$

Weighted Kernel Ridge Regression

- Consider the lth component of the function we wish to approximate. We will find a (possibly infinite) combination $\psi_l \in \mathcal{H}$ of features in $\mathcal{H}$ to represent the function

$$\hat{y}(l) = \langle \Phi(\xi), \psi_l \rangle_{\mathcal{H}} \approx f_l(x, u)$$

- For ease of notation, let

$$\underline{y}(l) \triangleq \begin{bmatrix} y_1(l) \\ \vdots \\ y_m(l) \end{bmatrix} \in \mathbb{R}^m \quad \langle \boldsymbol{\Phi}, \psi_l \rangle_{\mathcal{H}} \triangleq \begin{bmatrix} \langle \Phi(\xi_1), \psi_l \rangle_{\mathcal{H}} \\ \vdots \\ \langle \Phi(\xi_m), \psi_l \rangle_{\mathcal{H}} \end{bmatrix} \in \mathbb{R}^m$$

- Then we pose the weighted linear regression problem in $\mathcal{H}$ for the lth component of f

$$\psi_l^* = \underset{\psi_l \in \mathcal{H}}{\operatorname{argmin}} \left\| \sqrt{W^i} \left(\underline{y}(l) - \langle \boldsymbol{\Phi}, \psi_l \rangle_{\mathcal{H}} \right) \right\|_2^2 + \lambda \langle \psi_l, \psi_l \rangle_{\mathcal{H}}$$

- We introduce the "Ridge" penalty $\lambda \langle \psi_l, \psi_l \rangle_{\mathcal{H}}$ to avoid overfitting the training data when there are an excess of nonlinear features

Learning Subspace Property (LSP)

- A.K.A. "Occam's razor"
  - While there may be infinitely many $\psi_l \in \mathcal{H}$ satisfying the criteria, we look for the smallest $\psi_l \in span\{\Phi(\xi_j)\}_{j=1}^{m} = \mathcal{R}(\boldsymbol{\Phi})$ in the span of the training data
  - Analogous to right pseudoinverse of $\boldsymbol{\Phi}^*$ in $\mathcal{H}$
- In the case of *ridge* regression, we can actually prove that $\psi_l^* \in \mathcal{R}(\boldsymbol{\Phi})$

Suppose that $\psi_l^* = \hat{\psi}_l + \tilde{\psi}_l$ where $\hat{\psi}_l \in \mathcal{R}(\boldsymbol{\Phi})$ and $\tilde{\psi}_l \in \mathcal{R}(\boldsymbol{\Phi})^{\perp}$

Then $\langle \boldsymbol{\Phi}, \psi_l^* \rangle_{\mathcal{H}} = \langle \boldsymbol{\Phi}, \hat{\psi}_l \rangle_{\mathcal{H}} + \langle \boldsymbol{\Phi}, \tilde{\psi}_l \rangle_{\mathcal{H}} = \langle \boldsymbol{\Phi}, \hat{\psi}_l \rangle_{\mathcal{H}}$

But $\langle \psi_l^*, \psi_l^* \rangle_{\mathcal{H}} = \langle \hat{\psi}_l, \hat{\psi}_l \rangle_{\mathcal{H}} + \langle \tilde{\psi}_l, \tilde{\psi}_l \rangle_{\mathcal{H}} \geq \langle \hat{\psi}_l, \hat{\psi}_l \rangle_{\mathcal{H}}$

$$\therefore \quad \psi_l^* = \hat{\psi}_l \quad and \quad \exists a_l \in \mathbb{R}^m \quad s.t. \quad \psi_l^* = \boldsymbol{\Phi} a_l$$

- The regression problem now becomes finite-dimensional

Weighted Kernel Ridge Regression

- Use LSP to express $\psi_l = \mathbf{\Phi} a_l = a_{l_1}\Phi(\xi_1) + \cdots + a_{l_m}\Phi(\xi_m)$

$$a_l^* = \underset{a_l \in \mathbb{R}^m}{\operatorname{argmin}} \left\| \sqrt{W^i} \left(\underline{y}(l) - \sum_{j=1}^{m} \langle \mathbf{\Phi}, \Phi(\xi_j) \rangle_{\mathcal{H}} a_{l_j} \right) \right\|_2^2 + \lambda \sum_{i,j=1}^{m} a_{l_i} a_{l_j} \langle \Phi(\xi_i), \Phi(\xi_j) \rangle_{\mathcal{H}}$$

- Let $K \triangleq \langle \mathbf{\Phi}, \mathbf{\Phi} \rangle_{\mathcal{H}} \triangleq \left[\left\langle \Phi(\xi_i), \Phi(\xi_j) \right\rangle_{\mathcal{H}} \right]_{1 \le i,j \le m} \in \mathbb{R}^{m \times m}$ and do some simple manipulation

$$a_l^* = \underset{a_l \in \mathbb{R}^m}{\operatorname{argmin}} \left\| \sqrt{W^i} \left(\underline{y}(l) - K a_l \right) \right\|_2^2 + \lambda a_l^T K a_l$$

$$a_l^* = \underset{a_l \in \mathbb{R}^m}{\operatorname{argmin}} \left\| \begin{bmatrix} \sqrt{W^i} \underline{y}(l) \\ 0 \end{bmatrix} - \begin{bmatrix} \sqrt{W^i} K \\ \sqrt{\lambda K} \end{bmatrix} a_l \right\|_2^2$$

- The above is now a standard m-dimensional least squares problem whose solution is

$$a_l^* = \left(W^i K + \lambda I_m \right)^{-1} W^i \underline{y}(l)$$

$$\therefore \quad \hat{y}(l) = \langle \Phi(\xi), \psi_l^* \rangle_{\mathcal{H}} = \langle \Phi(\xi), \mathbf{\Phi} \rangle_{\mathcal{H}} \left(W^i K + \lambda I_m \right)^{-1} W^i \underline{y}(l)$$

Weighted Kernel Ridge Regression

- Notice that the entire solution to the regression problem is expressed in terms of inner products of elements in $\mathcal{H}$
  - Introduce kernel function $k(\xi, \eta) \triangleq \langle \Phi(\xi), \Phi(\eta) \rangle_{\mathcal{H}} \in \mathbb{R}$ and $\boldsymbol{k}(\xi) \triangleq \langle \boldsymbol{\Phi}, \Phi(\xi) \rangle_{\mathcal{H}} \in \mathbb{R}^m$
- Then the full estimator for the function is given by

$$\hat{y} = f^i(x, u) = \boldsymbol{Y} W^i \left(K W^i + \lambda I_m \right)^{-1} \boldsymbol{k}(\xi) \quad K = \left[k\left(\xi_i, \xi_j \right) \right]_{1 \le i,j \le m} \quad \xi = \begin{bmatrix} x \\ u \\ 1 \end{bmatrix}$$

- This approximation is easily differentiated if we know the derivative of the kernel with respect to one of its slots $D_\xi k(\xi_0, \xi)$ allowing us to find $D_\xi \boldsymbol{k}(\xi_0) \in \mathbb{R}^{m \times m}$

$$D_\xi f^i(\xi_0) \Delta\xi = \boldsymbol{Y} W^i \left(K W^i + \lambda I_m \right)^{-1} D_\xi \boldsymbol{k}(\xi_0) \Delta\xi$$

Kernel Regression Implementation

- It is prohibitively expensive to invert the $\left(KW^i + \lambda I_m\right) \in \mathbb{R}^{m \times m}$ each time the models are updated
- We therefore choose only the largest $m' < m$ weights $\left\{\bar{Q}^i_{j_1}, \bar{Q}^i_{j_2}, \ldots, \bar{Q}^i_{j_{m'}}\right\}$ on training points $S^i = \{j_1, j_2, \ldots, j_{m'}\}$ to be nonzero. ➔ "Restricted EM"
  - With the intention of doing *local* modeling, this is a sensible thing to do to speed up the algorithm
- Forming the sub-matrices

$$\widetilde{K} = \left[K_{i_r j_s}\right]_{1 \le r,s \le m'} \quad \widetilde{W} = diag\left\{\bar{Q}^i_{j_r}\right\}_{1 \le r \le m} \quad \widetilde{\boldsymbol{Y}} = \left[y_{j_1} \cdots y_{j_{m'}}\right]$$

$$\widetilde{\boldsymbol{k}}(\xi) = \left[\left\langle \Phi\left(\xi_{j_1}\right), \psi_l \right\rangle_{\mathcal{H}} \quad \cdots \quad \left\langle \Phi\left(\xi_{j_{m'}}\right), \psi_l \right\rangle_{\mathcal{H}}\right]^T$$

Using the Schur complement, we find that

$$\hat{y} = f^i(x,u) = \widetilde{\boldsymbol{Y}} \widetilde{W}^i \left(\widetilde{K} \widetilde{W}^i + \lambda I_{m'}\right)^{-1} \widetilde{\boldsymbol{k}}(\xi)$$

Application to Duffing Equation with Control (Restricted EM Algorithm)

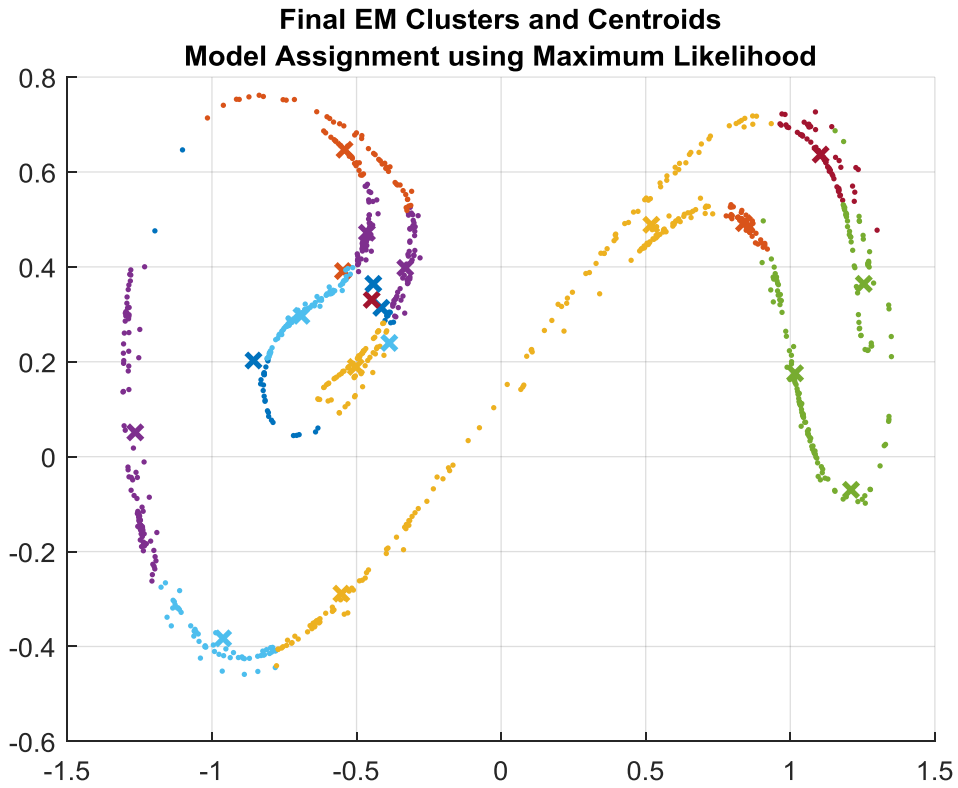
- $N = 20$ models were fit using the RBF kernel with $\sigma = 0.1$ and small $\lambda = 5 * 10^{-4}$

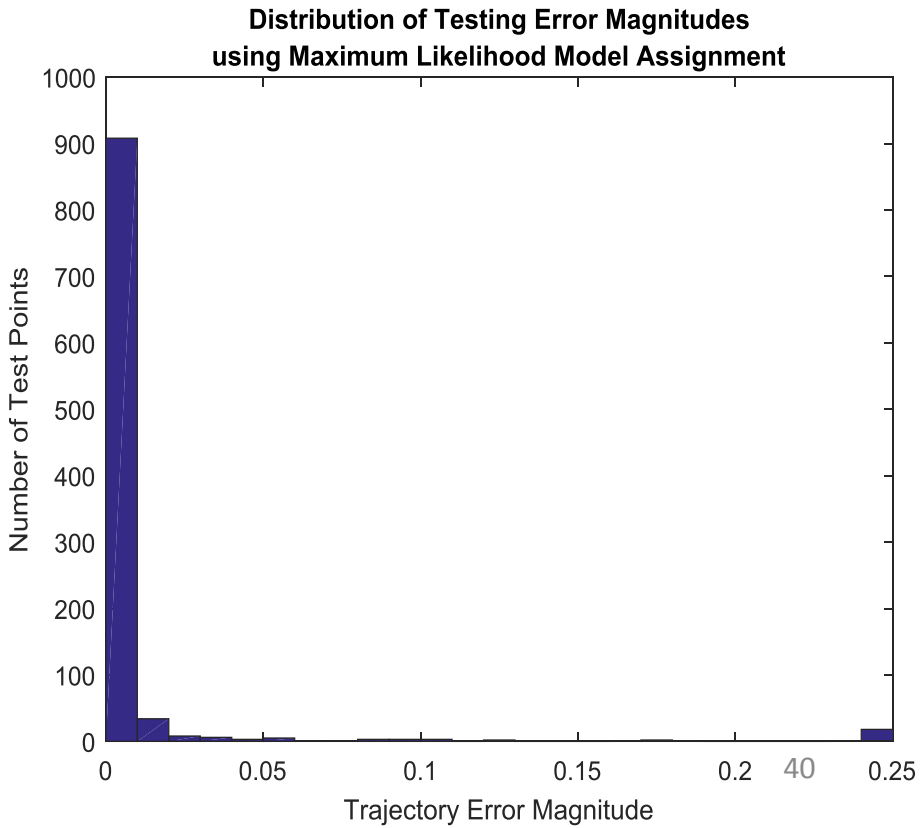
$$\boldsymbol{k}(x, z) = \exp\left(-\frac{1}{2\sigma^2}\|x - z\|_2^2\right)$$

- Each model is fit to the $m' = 500$ training points with the largest weights $\bar{Q}_j^i$
- Maximum likelihood model assignment was used to reconstruct the dynamics
  - Numerical experiments show that of the methods discussed, maximum likelihood produced the lowest errors.
- Minimum singular values of covariances Σ and R were needed to prevent degeneracy
- $m_{test} = 1000$ independently generated test points with random control perturbations were used for evaluation

Nonlinear Restricted EM Results

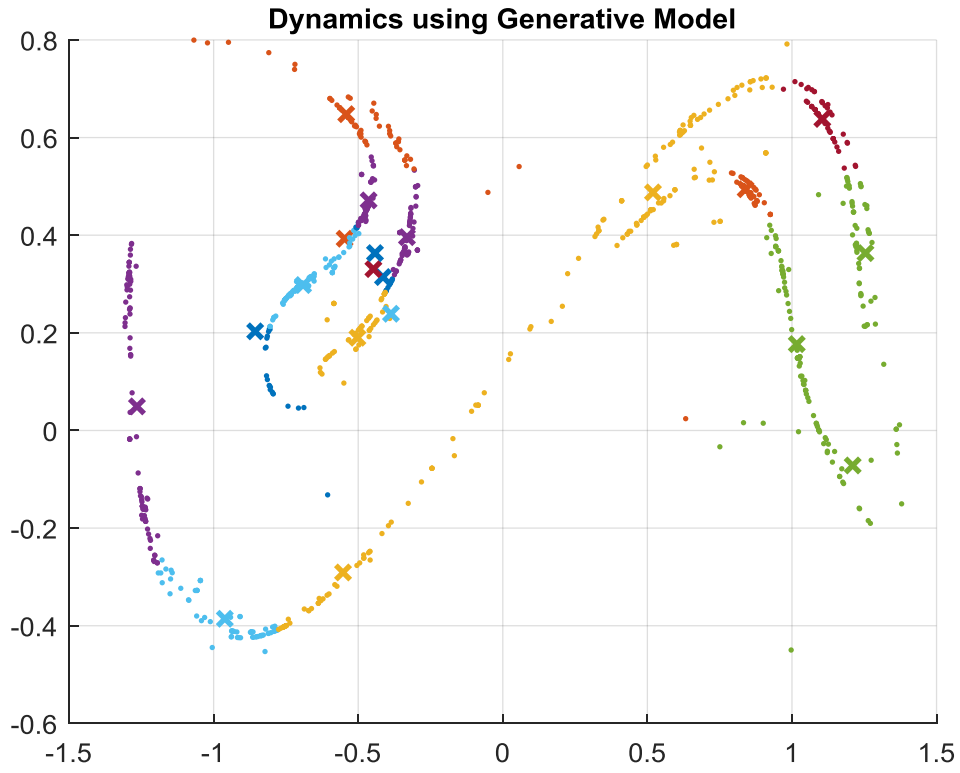
- Mean Distance Error: $\boxed{\bar{E}_D = 0.0209}$
- Median Distance Error: 0.0014
- Decaying Error Norm over $M = 10$ steps with $\alpha = 2$: $\boxed{\bar{E}_\alpha^M = 0.1196}$

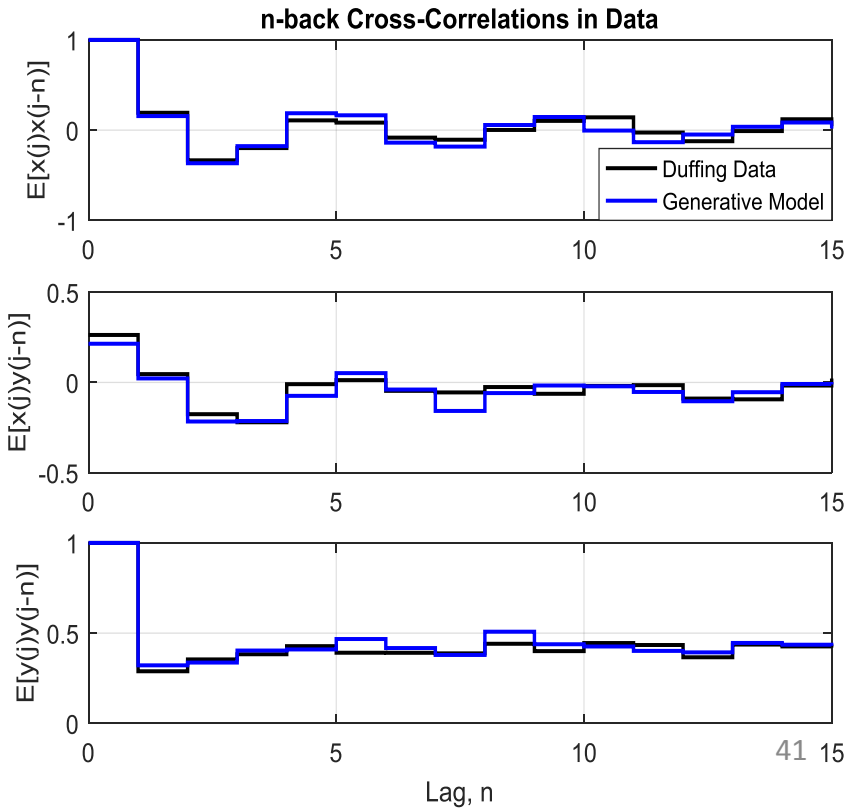




Use as a Generative Model

- The dynamics were simulated using the learned model for 1000 steps with no control perturbations
- The autocorrelation was compared to the actual Poincare map





Application to Duffing Equation with Control (K-Means Algorithm)

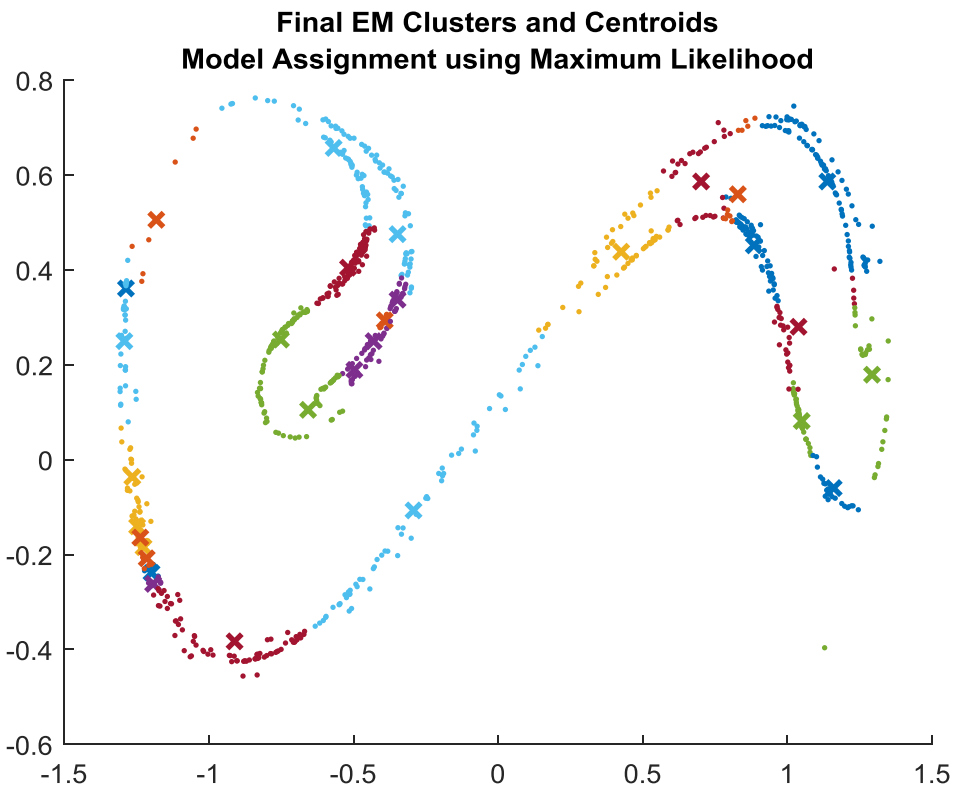
- $N = 30$ models were fit using the RBF kernel with $\sigma = 0.1$ and small $\lambda = 5 * 10^{-4}$

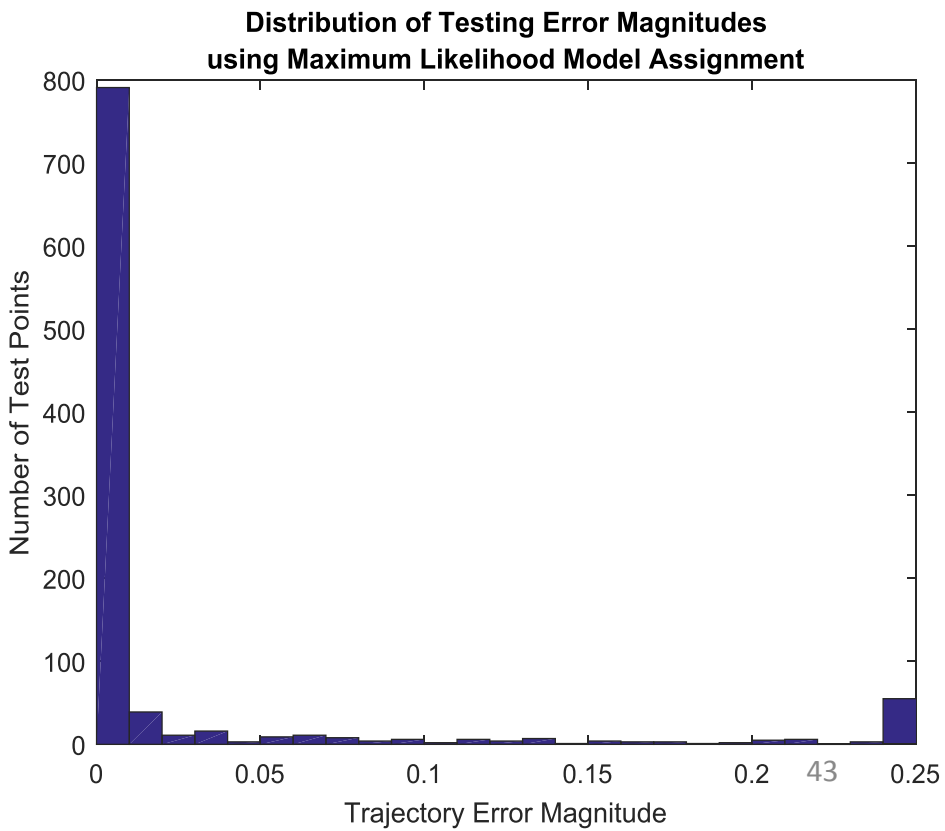
$$\boldsymbol{k}(x, z) = \exp\left(-\frac{1}{2\sigma^2}\|x - z\|_2^2\right)$$

- Each training point was assigned to a single model based on maximum likelihood giving sparse $\bar{Q}_j^i$
  - Convergence was much faster even though more models were fit
- Maximum likelihood model assignment was used to reconstruct the dynamics
- Minimum singular values of covariances Σ and R were needed to prevent degeneracy
- $m_{test} = 1000$ independently generated test points with random control perturbations were used for evaluation

Nonlinear K-Means Results

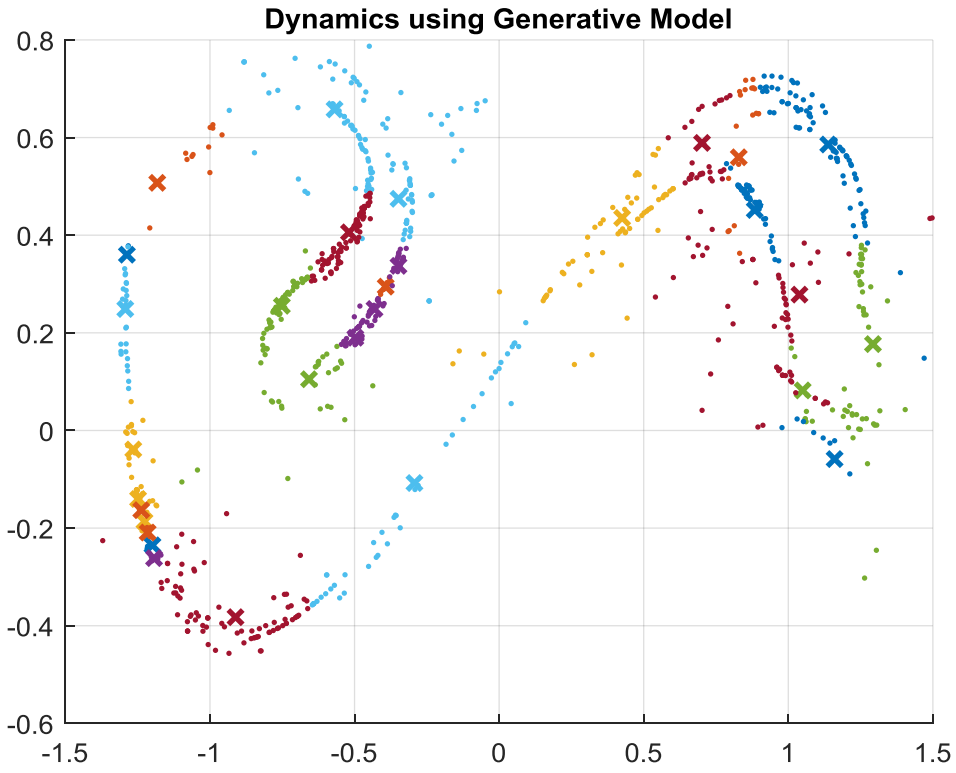
- Mean Distance Error: $\boxed{\bar{E}_D = 0.0500}$
- Median Distance Error: 0.0021
- Decaying Error Norm over $M = 10$ steps with $\alpha = 2$: $\boxed{\bar{E}_\alpha^M = 0.2765}$

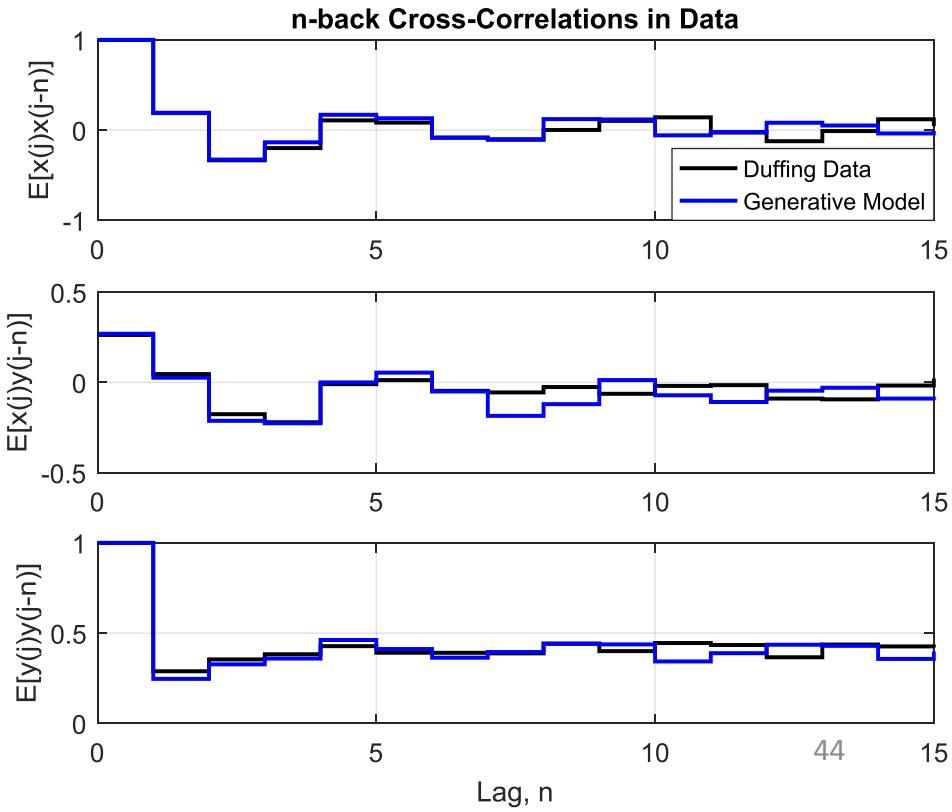




Use as a Generative Model

- The dynamics were simulated using the learned model for 1000 steps with no control perturbations
- The autocorrelation was compared to the actual Poincare map





Application to Duffing Equation with Control (Generalized K-Means Algorithm)

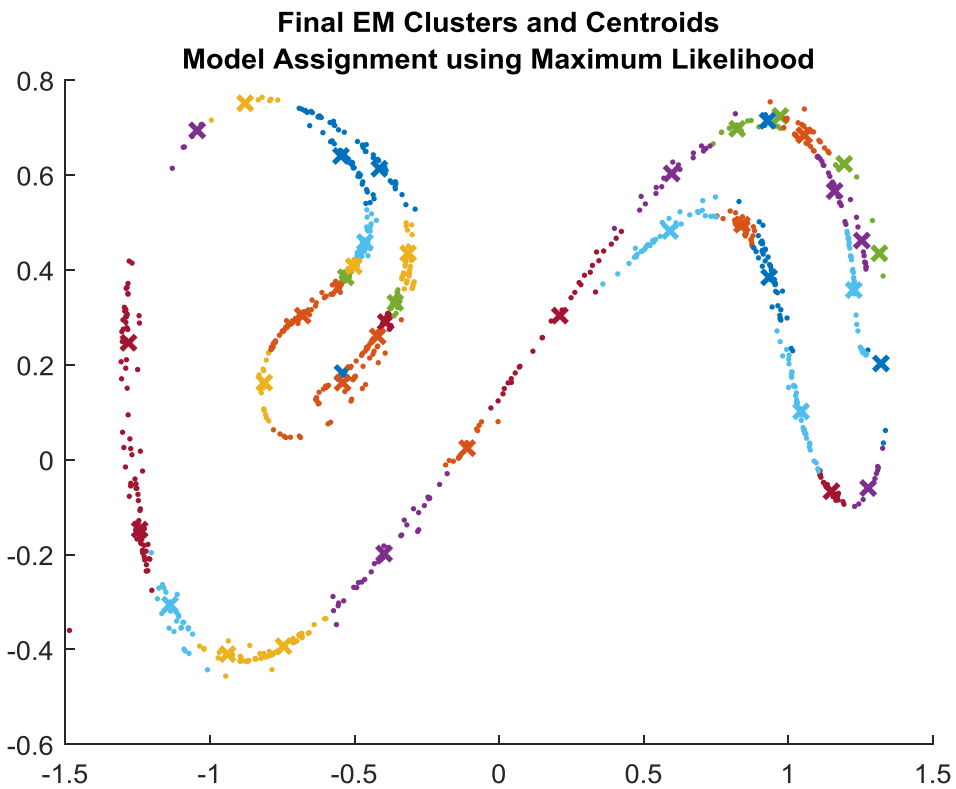
- $N = 40$ models were fit using the RBF kernel with $\sigma = 0.1$ and small $\lambda = 5 * 10^{-4}$

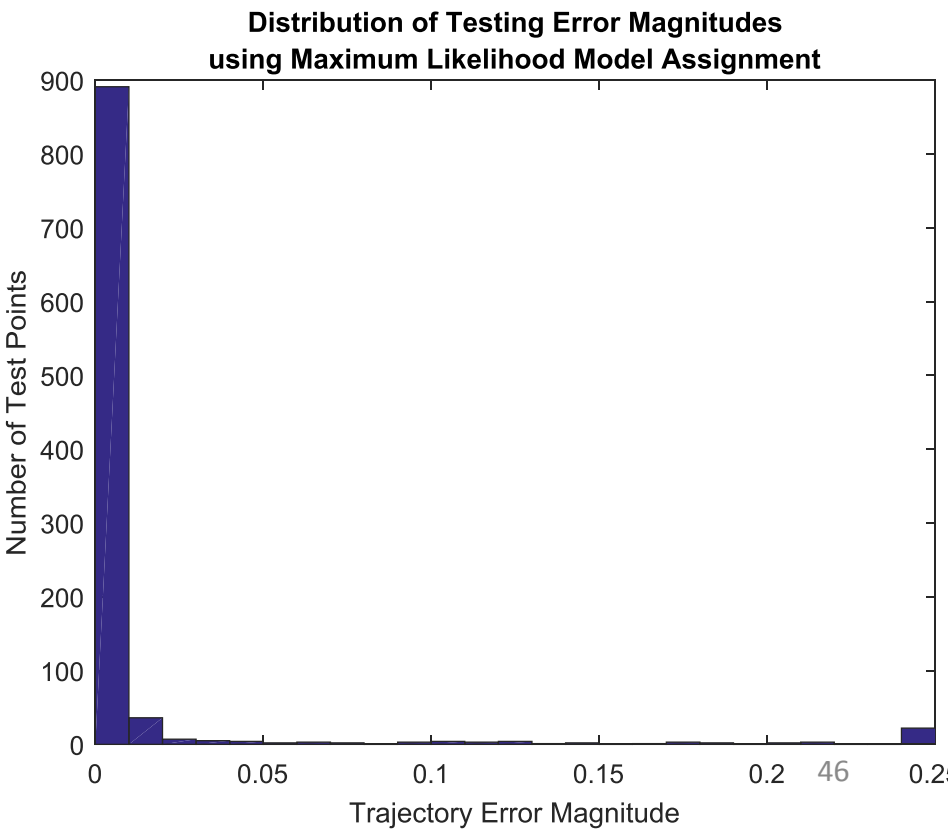
$$\boldsymbol{k}(x, z) = \exp\left(-\frac{1}{2\sigma^2}\|x - z\|_2^2\right)$$

- Each training point was assigned to $N' < N$ models based on maximum likelihood giving sparse $\bar{Q}_j^i$
- K-means was initially converged with $N' = 1$ then increased to $N' = 4$ and converged
  - This encourages the centroids to stay tight to the filamented distribution of points in the Poincare map
- Maximum likelihood model assignment was used to reconstruct the dynamics
- Minimum singular values of covariances Σ and R were needed to prevent degeneracy
- $m_{test} = 1000$ independently generated test points with random control perturbations were used for evaluation

Nonlinear Generalized K-Means Results

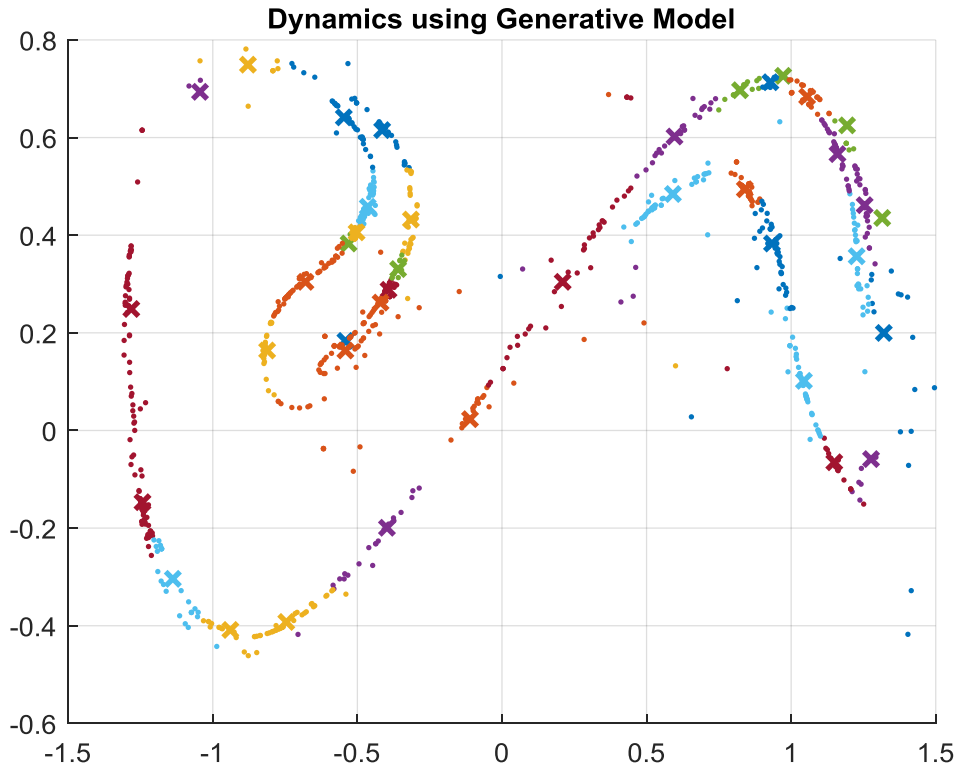
- Mean Distance Error: $\boxed{\bar{E}_D = 0.0208}$
- Median Distance Error: 0.0017
- Decaying Error Norm over $M = 10$ steps with $\alpha = 2$: $\boxed{\bar{E}_\alpha^M = 0.1434}$

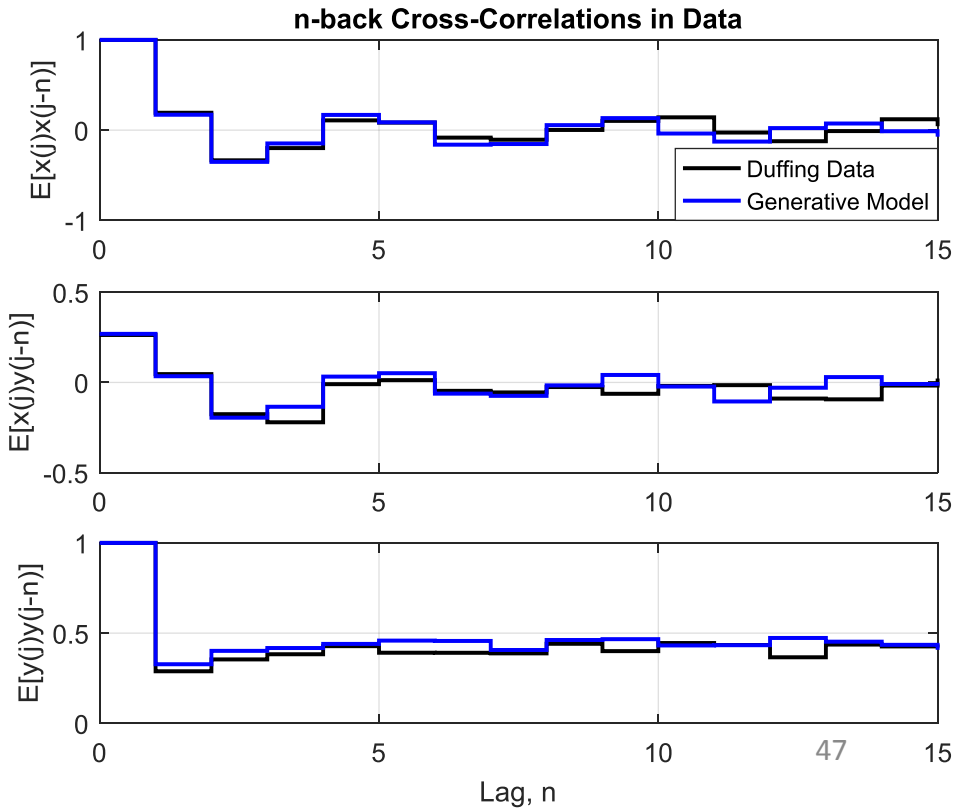




Use as a Generative Model

- The dynamics were simulated using the learned model for 1000 steps with no control perturbations
- The autocorrelation was compared to the actual Poincare map





Highlights from Nonlinear Cases

- The Restricted EM algorithm has the lowest error with the fewest number of models.
  - The algorithm takes a long time to converge
  - Centroids are not tight to filamented point distribution
- K-means converges quickly when points are assigned to only a single model.
  - Higher error
  - More models are required
  - Centroids are tight to filamented point distribution
- Generalized K-means where points are assigned to multiple models improves the error over K-means.
  - Many models are required
  - Low error
  - Centroids are tight to filamented point distribution

Non-Gaussian Local Distributions

- We have shown that fitting nonlinear kernel models allows us to better capture the dynamics in our model
- However, The Gaussian densities themselves do not reflect the local curvature and filamentation of the Chaotic Poincare map
  - Therefore, many models are needed
- It would be helpful to allow the local distributions $p_{X|Z}(x|Z)$ to become curved and stretched
- This is a work in progress, but it can accomplished by fitting Gaussian densities in a kernel Hilbert space

$$p_{X|Z}(x|Z) = C \exp\left[-\frac{1}{2}\left\langle \tilde{\Sigma}(\Phi(x) - \mu_\Phi), (\Phi(x) - \mu_\Phi)\right\rangle_{\mathcal{H}}\right]$$

$$\mu_\Phi = \frac{1}{|S^i|}\sum_{j \in S^i} \Phi(x_j)$$

Gaussians in Feature Space

- Roughly the idea is to find the covariance of the data $\{x_j\}_{j=1}^m$ in feature space $\mathcal{H}$ where

$$\boldsymbol{\Phi}: \mathbb{R}^m \to \mathcal{H}, \qquad v \mapsto v_1 \Phi(x_1) + \cdots + v_m \Phi(x_m)$$

$$\Sigma = \frac{1}{m}\left(\boldsymbol{\Phi} - \frac{1}{m}\boldsymbol{\Phi}\mathbf{1}_m \mathbf{1}_m^T\right)\left(\boldsymbol{\Phi} - \frac{1}{m}\boldsymbol{\Phi}\mathbf{1}_m \mathbf{1}_m^T\right)^* : \mathcal{H} \to \mathcal{H}$$

$$\Sigma = \frac{1}{m}\boldsymbol{\Phi}\left(I_m - \frac{1}{m}\mathbf{1}_m \mathbf{1}_m^T\right)\boldsymbol{\Phi}^*$$

- Then guess the form of the inverse operator $\tilde{\Sigma}$ restricted to the learning subspace $\mathcal{R}(\boldsymbol{\Phi})$
  - We know that its domain and range are $\mathcal{R}(\boldsymbol{\Phi})$
  - We know that it is self-adjoint

$$\tilde{\Sigma} = \boldsymbol{\Phi} B \boldsymbol{\Phi}^* \quad \text{for some} \quad B = B^T \in \mathbb{R}^{m \times m}$$

Gaussians in Feature Space

- In order for $\tilde{\Sigma} = \boldsymbol{\Phi} B \boldsymbol{\Phi}^*$ to be the inverse of $\Sigma = \frac{1}{m}\boldsymbol{\Phi}\left(I_m - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^T\right)\boldsymbol{\Phi}^*$ over the learning subspace, we must have

$$K = \boldsymbol{\Phi}^*\boldsymbol{\Phi} = \boldsymbol{\Phi}^*\tilde{\Sigma}\Sigma\boldsymbol{\Phi} = \frac{1}{m}\underbrace{\boldsymbol{\Phi}^*\boldsymbol{\Phi}}_{K} B \underbrace{\boldsymbol{\Phi}^*\boldsymbol{\Phi}}_{K}\left(I_m - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^T\right)\underbrace{\boldsymbol{\Phi}^*\boldsymbol{\Phi}}_{K}$$

And

$$K = \boldsymbol{\Phi}^*\boldsymbol{\Phi} = \boldsymbol{\Phi}^*\Sigma\tilde{\Sigma}\boldsymbol{\Phi} = \frac{1}{m}\underbrace{\boldsymbol{\Phi}^*\boldsymbol{\Phi}}_{K}\left(I_m - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^T\right)\underbrace{\boldsymbol{\Phi}^*\boldsymbol{\Phi}}_{K} B \underbrace{\boldsymbol{\Phi}^*\boldsymbol{\Phi}}_{K}$$

- We must have

$$\boxed{B = m\left[K\left(I_m - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^T\right)K\right]^+}$$

- Therefore, the non-normalized density is given by

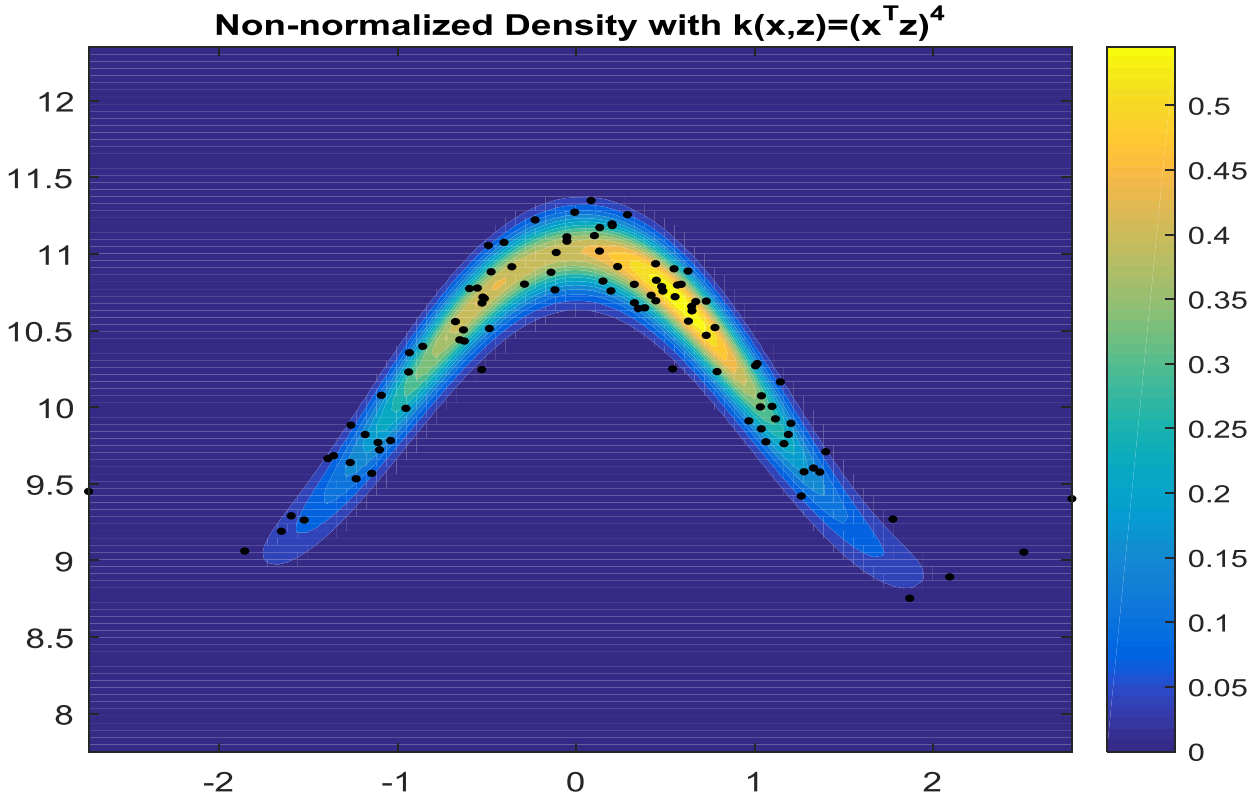
$$p_{X|Z}(x|Z) = C\exp\left[-\frac{1}{2}\left(\Phi(x) - \frac{1}{m}\boldsymbol{\Phi}\mathbf{1}_m\right)^* \boldsymbol{\Phi} B \boldsymbol{\Phi}^*\left(\Phi(x) - \frac{1}{m}\boldsymbol{\Phi}\mathbf{1}_m\right)\right]$$

$$\boxed{p_{X|Z}(x|Z) = C\exp\left[-\frac{1}{2}\left(\boldsymbol{k}(x) - \frac{1}{m}K\mathbf{1}_m\right)^* B\left(\boldsymbol{k}(x) - \frac{1}{m}K\mathbf{1}_m\right)\right]}$$

- I don't know of a good direct way to find C, but we do have a very convenient Monte-Carlo sample $\{x_1, \ldots, x_m\}$ to approximate it!

Gaussians in Feature Space

- (Macready, W. G., "Density Estimation with Mercer Kernels," NASA Ames Research Center) presents an EM algorithm for fitting mixtures of Gaussians in nonlinear feature space.
- The following figure was made using the method described above with $m = 100$ points and a 4^{th} order polynomial kernel. The RBF kernel tends to over-fit.



Many Avenues for Future Work

- Use linear models to perform local DMD in identified regions of interest
  - Spectral analysis is also possible using the kernel model's Jacobian $D_{\xi} f^{i}(\xi_0)$
- Identifying global topological structure
  - Maximum likelihood model classification partitions the space efficiently ➔ Look at Markov transition probabilities between models
  - Patch together Gaussian "pancakes" to study the underlying data manifold ➔ non-parametric manifold learning (look at Lukaszyk-Karmowski metric)
  - Help identify periodic orbits, tori, and attractors

Many Avenues for Future Work

- Use linear models and conditional probabilities to do state estimation
  - Account for noisy observation process
  - Develop Kalman filter
- Possibility for online formulation
  - New data can be used to update models using matrix inversion lemma
  - K-means clustering can be performed “greedily”
  - Application to streaming DMD
- Given that both parts (Regression and Gaussian Mixture fitting) of the maximization step can be kernelized suggests that *the entire algorithm might be performed in high-dimensional intrinsic space and kernelized!*

Many Avenues for Future Work

- Investigate how the algorithm behaves for high-dimensional systems
  - Dealing with degenerate Gaussians and dynamics
  - Learning data manifold and locally projecting onto it
  - Investigate dimensionality reduction (kernel PCA or autoencoder) prior to model fitting
- Continuous reconstruction of the model estimates also require further investigation
  - Need some kind of probability-weighted smooth interpolation
- Implementation note: Since each model in the Maximization step is fit independently, the algorithm may be executed in parallel.