An EM Algorithm for Learning Local Dynamics Models In a Bayesian Framework

Samuel Otto

Introduction

Derivation of Algorithm

Suppose we have some nonlinear function of the following form

$$y = f(x,u), \quad x, y \in \mathbb{R}^n, \quad u \in \mathbb{R}^q$$

We have in mind a discrete or continuous time dynamical system with control input u. Suppose we have collected training data $\{(x_j, u_j, y_j)\}_{j=1}^{m}$. We wish to approximate the function f by a collection of affine linear models spread throughout the domain. In addition to the models themselves, we require a way to assign a new point x to the nearby model which will best approximate its dynamics under f. Let there be N linear models $\mathcal{M} = \{M^i = (A^i, B^i, \mu_x^i, \mu_y^i, R^i, \Sigma^i, \phi^i)\}_{i=1}^{N}$ of the form

$$y = \mu_y^i + A^i(x - \mu_x^i) + B^i u + v^i, \quad v^i \sim \mathcal{N}(0, R^i), \quad i = 1, \dots, N$$

where v^i is Gaussian noise having covariance R^i. Let z_j be the random variable whose value is the model describing the dynamics at the training point x_j, u_j. Assume the probability density that the dynamics of the point x_j are modeled by the ith linear model is also Gaussian in the spatial coordinates

$$P(x_j | z_j = i) = \mathcal{N}(\mu_x^i, \Sigma^i)(x_j)$$

Assume that the prior probabilities are constants

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

We wish to find the parameters of $\mathcal{M}$ which maximize the likelihood of the training data

$$L(\mathcal{M}) = \prod_{j=1}^{m} P(y_j, x_j, u_j; \mathcal{M})$$

We know from our assumptions that

$$P(y_j, x_j, u_j; \mathcal{M}) = \sum_{i=1}^{N} P(y_j, x_j, u_j, z_j = i; \mathcal{M})$$

$$P(y_j, x_j, u_j; \mathcal{M}) = \sum_{i=1}^{N} \underbrace{P(y_j | (x_j, u_j, z_j = i); \mathcal{M})}_{P(v_j^i; \mathcal{M})} P(x_j, u_j | z_j = i; \mathcal{M}) P(z_j = i; \mathcal{M})$$

$$P(y_j, x_j, u_j; \mathcal{M}) = \sum_{i=1}^{N} \mathcal{N}(0, R^i)(v_j^i)\mathcal{N}(\mu_x^i, \Sigma^i)(x_j)\phi^i$$

Therefore, the problem is in a form to which the EM algorithm is directly applicable. We quickly review the EM algorithm as it applies to the problem of maximizing the above likelihood function. Find the log likelihood

$$l(\mathcal{M}) = \sum_{j=1}^{m} \log\left[\sum_{i=1}^{N} P(v_j^i)P(x_j|z_j = i)P(z_j = i)\right]$$

Introduce a collection of yet to be determined distributions over the possible models for each training point $Q_j(z_j)$.

$$l(\mathcal{M}) = \sum_{j=1}^{m} \log\left[\sum_{i=1}^{N} Q_j(z_j = i)\frac{P(v_j^i)P(x_j|z_j = i)P(z_j = i)}{Q_j(z_j = i)}\right]$$

The inside term is now in the form of an expectation

$$l(\mathcal{M}) = \sum_{j=1}^{m} \log\left(E_{z_j \sim Q_j}\left[\frac{P(v_j^i)P(x_j|z_j)P(z_j)}{Q_j(z_j)}\right]\right)$$

By Jensen's inequality and the concavity of the logarithm, we observe that

$$l(\mathcal{M}) \geq \sum_{j=1}^{m} E_{z_j \sim Q_j}\left[\log\left(\frac{P(v_j^i)P(x_j|z_j)P(z_j)}{Q_j(z_j)}\right)\right]$$

$$l(\mathcal{M}) \geq \sum_{j=1}^{m}\sum_{i=1}^{N} Q_j(z_j = i)\log\left(\frac{P(v_j^i; \mathcal{M})P(x_j|z_j = i; \mathcal{M})P(z_j = i; \mathcal{M})}{Q_j(z_j = i)}\right)$$

Substitute the known forms of the distributions

$$\boxed{l(\mathcal{M}) \geq \sum_{j=1}^{m}\sum_{i=1}^{N} Q_j(z_j = i)\log\left(\frac{\mathcal{N}(0, R^i)(v_j^i)\mathcal{N}(\mu_x^i, \Sigma^i)(x_j)\phi^i}{Q_j(z_j = i)}\right)}$$

$$v_j^i = y_j - \left[\mu_y^i + A^i(x_j - \mu_x^i) + B^i u_j\right]$$

Let $\bar{\mathcal{M}}$ be a previous set of parameters for the collection of linear models. The distribution $Q_j(z_j)$ is set to be the normalized probability

$$Q_j(z_j = i) = \frac{P(y_j, x_j, u_j, z_j = i; \mathcal{M})}{\sum_{k=1}^{N} P(y_j, x_j, u_j, z_j = k; \mathcal{M})} = \frac{P(v_j^i; \bar{\mathcal{M}})P(x_j|z_j = i; \bar{\mathcal{M}})P(z_j = i; \bar{\mathcal{M}})}{\sum_{k=1}^{N} P(v_j^k; \bar{\mathcal{M}})P(x_j|z_j = k; \bar{\mathcal{M}})P(z_j = k; \bar{\mathcal{M}})}$$

$$\boxed{Q_j(z_j = i) = \frac{\mathcal{N}(0,\bar{R}^i)(v_j^i)\mathcal{N}(\bar{\mu}_x^i,\bar{\Sigma}^i)(x_j)\bar{\phi}^i}{\sum_{k=1}^N \mathcal{N}(0,\bar{R}^k)(v_j^k)\mathcal{N}(\bar{\mu}_x^k,\bar{\Sigma}^k)(x_j)\bar{\phi}^k}}$$

This choice is made so that the above inequality becomes equality if and only if $\mathcal{M} = \bar{\mathcal{M}}$. One can see that this must be the case since the term inside the expectation becomes independent of z_j when $\mathcal{M} = \bar{\mathcal{M}}$ and therefore $\log\left(E_{z_j\sim Q_j}[\cdots]\right) = E_{z_j\sim Q_j}[\log(\cdots)]$. The EM algorithm proceeds by repeating the following two steps until convergence

E-step: Set the nominal value $\bar{\mathcal{M}} \coloneqq \mathcal{M}$ and find

$$\bar{Q}_j^i \triangleq Q_j(z_j = i; \bar{\mathcal{M}}) = \frac{\mathcal{N}(0,\bar{R}^i)(v_j^i)\mathcal{N}(\bar{\mu}_x^i,\bar{\Sigma}^i)(x_j)\bar{\phi}^i}{\sum_{k=1}^N \mathcal{N}(0,\bar{R}^k)(v_j^k)\mathcal{N}(\bar{\mu}_x^k,\bar{\Sigma}^k)(x_j)\bar{\phi}^k}$$

M-step: Maximize the lower bound on the likelihood using the above expectation

$$\mathcal{M} \coloneqq \underset{\mathcal{M}}{\operatorname{argmax}} \sum_{j=1}^m \sum_{i=1}^N \bar{Q}_j^i \log\left(\frac{\mathcal{N}(0,R^i)(v_j^i)\mathcal{N}(\mu_x^i,\Sigma^i)(x_j)\phi^i}{\bar{Q}_j^i}\right)$$

Subject to the constraint

$$1 = \sum_{i=1}^N \phi^i$$

Let us proceed with the maximization by splitting up the objective

$$\mathcal{M} \coloneqq \underset{\mathcal{M}}{\operatorname{argmax}} \underbrace{\sum_{j=1}^m \sum_{i=1}^N \bar{Q}_j^i \log\left(\mathcal{N}(0,R^i)(v_j^i)\right)}_{(I)} + \underbrace{\sum_{j=1}^m \sum_{i=1}^N \bar{Q}_j^i \log\left(\mathcal{N}(\mu_x^i,\Sigma^i)(x_j)\right)}_{(II)} + \underbrace{\sum_{j=1}^m \sum_{i=1}^N \bar{Q}_j^i \log(\phi^i)}_{(III)}$$

Maximizing the second term (II) is analogous to the well-known mixture of Gaussians model so we will carry out this maximization first

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

Solve for the centroids of the spatial Gaussians

$$D_{\mu_x^k}(II)(h) = 0 = \sum_{j=1}^m \frac{1}{2}\bar{Q}_j^k (x_j - \mu_x^k)^T (\Sigma^k)^{-1} h \quad \forall h \in \mathbb{R}^n$$

$$\sum_{j=1}^m \bar{Q}_j^k x_j = \sum_{j=1}^m \bar{Q}_j^k \mu_x^k \Rightarrow \boxed{\mu_x^k = \frac{\sum_{j=1}^m \bar{Q}_j^k x_j}{\sum_{j=1}^m \bar{Q}_j^k}}$$

Solve for the spatial covariance matrices.

$$D_{\Sigma^k}(II)(H) = 0 = \sum_{j=1}^{m} \bar{Q}_j^k \left[-\frac{1}{2}\frac{1}{|\Sigma^k|}\left|\Sigma^k\right| tr\left({\Sigma^k}^{-1}H\right) + \frac{1}{2}\left(x_j - \mu_x^k\right)^T \left(\Sigma^k\right)^{-1} H \left(\Sigma^k\right)^{-1}\left(x_j - \mu_x^k\right) \right]$$

$$0 = \sum_{j=1}^{m} \bar{Q}_j^k \left[\left(x_j - \mu_x^k\right)^T \left(\Sigma^k\right)^{-1} H \left(\Sigma^k\right)^{-1}\left(x_j - \mu_x^k\right)\right] - tr\left({\Sigma^k}^{-1}H\right)\sum_{j=1}^{m} \bar{Q}_j^k \qquad \forall H \in \mathbb{R}^{n\times n}$$

Let

$$S^k = \sum_{j=1}^{m} \bar{Q}_j^k \left(x_j - \mu_x^k\right)\left(x_j - \mu_x^k\right)^T$$

Then

$$0 = tr\left(S^k \left(\Sigma^k\right)^{-1} H \left(\Sigma^k\right)^{-1}\right) - tr\left({\Sigma^k}^{-1}H\right)\sum_{j=1}^{m} \bar{Q}_j^k$$

$$0 = tr\left[\left(\Sigma^k\right)^{-1} S^k \left(\Sigma^k\right)^{-1} H - \left(\sum_{j=1}^{m} \bar{Q}_j^k\right)\left(\Sigma^k\right)^{-1} H\right] = \left\langle \left(\Sigma^k\right)^{-1} S^k \left(\Sigma^k\right)^{-1} - \left(\sum_{j=1}^{m} \bar{Q}_j^k\right)\left(\Sigma^k\right)^{-1}, H \right\rangle$$

$$\therefore \quad \left(\Sigma^k\right)^{-1} S^k \left(\Sigma^k\right)^{-1} = \left(\sum_{j=1}^{m} \bar{Q}_j^k\right)\left(\Sigma^k\right)^{-1}$$

$$S^k = \left(\sum_{j=1}^{m} \bar{Q}_j^k\right)\Sigma^k \Longrightarrow \boxed{\Sigma^k = \frac{\sum_{j=1}^{m} \bar{Q}_j^k \left(x_j - \mu_x^k\right)\left(x_j - \mu_x^k\right)^T}{\sum_{j=1}^{m} \bar{Q}_j^k}}$$

Now we will proceed to maximize the first term (I)

$$\underset{\mathcal{M}}{\operatorname{argmax}}\ (I) = \underset{\mathcal{M}}{\operatorname{argmax}} \sum_{j=1}^{m}\sum_{i=1}^{N} \bar{Q}_j^i \left[-\frac{1}{2}\log\left|R^i\right| - \frac{1}{2}\left(v_j^i\right)^T \left(R^i\right)^{-1}\left(v_j^i\right)\right]$$

$$v_j^i = y_j - \left[\mu_y^i + A^i\left(x_j - \mu_x^i\right) + B^i u_j\right]$$

The result from above immediately gives

$$\boxed{R^k = \frac{\sum_{j=1}^{m} \bar{Q}_j^k \left(v_j^k\right)\left(v_j^k\right)^T}{\sum_{j=1}^{m} \bar{Q}_j^k}}$$

We now must find the matrices A^k and B^k in addition to μ_y^i. This is the classic weighted linear regression problem with weights given by $\bar{Q}_j^k$. Let

$$\xi_j^k = \begin{bmatrix} x_j - \mu_x^k \\ u_j \\ 1 \end{bmatrix}, \quad \tilde{A}^k = \begin{bmatrix} A^k & B^k & \mu_y^k \end{bmatrix}$$

Then

$$v_j^i = y_j - \tilde{A}^k \xi_j^k$$

and we will try to find A^k, B^k and μ_y^i all at the same time by differentiating with respect to $\tilde{A}^k$

$$D_{\tilde{A}^k}(I)(H) = 0 = \sum_{j=1}^{m} \bar{Q}_j^k \left(v_j^k\right)^T \left(R^k\right)^{-1} H \xi_j^k \quad \forall H \in \mathbb{R}^{n\times(n+q)}$$

Let

$$S^k = \sum_{j=1}^{m} \bar{Q}_j^k \xi_j^k \left(v_j^k\right)^T$$

Then

$$0 = tr\left(S^k \left(R^k\right)^{-1} H\right) = \langle \left(R^k\right)^{-1} \left(S^k\right)^T, H \rangle \Rightarrow S^k \left(R^k\right)^{-1} = 0 \Rightarrow S^k = 0$$

$$\sum_{j=1}^{m} \bar{Q}_j^k \xi_j^k \left(y_j - \tilde{A}^k \xi_j^k\right)^T = 0 \Rightarrow \sum_{j=1}^{m} \bar{Q}_j^k \left(y_j - \tilde{A}^k \xi_j^k\right) \left(\xi_j^k\right)^T = 0$$

$$\sum_{j=1}^{m} \bar{Q}_j^k y_j \left(\xi_j^k\right)^T = \sum_{j=1}^{m} \bar{Q}_j^k \tilde{A}^k \xi_j^k \left(\xi_j^k\right)^T = \tilde{A}^k \left(\sum_{j=1}^{m} \bar{Q}_j^k \xi_j^k \left(\xi_j^k\right)^T\right)$$

$$\boxed{\tilde{A}^k = \begin{bmatrix} A^k & B^k & \mu_y^k \end{bmatrix} = \left(\sum_{j=1}^{m} \bar{Q}_j^k y_j \left(\xi_j^k\right)^T\right) \left(\sum_{j=1}^{m} \bar{Q}_j^k \xi_j^k \left(\xi_j^k\right)^T\right)^{-1}, \quad \xi_j^k = \begin{bmatrix} x_j - \mu_x^k \\ u_j \\ 1 \end{bmatrix}}$$

It should be noted that our observables x_j are not limited to be linear functions. By choosing to include quadratic and higher terms, we can build a mixture of nonlinear models instead of linear ones. The second order terms may be used to estimate the error due to bias in the model. Finally, we maximize the third term (III) subject to the constraint by forming the Lagrangian

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

$$\frac{\partial \mathcal{L}}{\partial \phi^k} = 0 = \sum_{j=1}^{m} \frac{\bar{Q}_j^k}{\phi^k} - \lambda \Longrightarrow \phi^k = \frac{1}{\lambda} \sum_{j=1}^{m} \bar{Q}_j^k$$

Solving for λ using the constraint

$$\lambda = \sum_{i=1}^{N} \sum_{j=1}^{m} \bar{Q}_j^i = \sum_{j=1}^{m} 1 = m$$

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

Reconstructing Estimates

Once the linear models are fit to the training data, the dynamics of a new point can be estimated using the collection of linear models and their densities in a number of ways. Here we will present three possibilities:

1. The simplest method is to choose the maximum likelihood linear model at x

$$\boxed{k^*(x) = \operatorname*{argmax}_{k=1,\dots,N} \phi^k \mathcal{N}\left(\mu_x^k, \Sigma^k\right)(x)}$$

$$\boxed{\hat{y}(x,u) = \mu_y^{k^*} + A^{k^*}\left(x - \mu_x^{k^*}\right) + B^{k^*} u}$$

2. The first method allows for approximations which are discontinuous. If we desire continuity then a natural method is to combine some or all linear models at a given point according to their likelihood. Hence,

$$\boxed{w^k(x) = P(z = k|x) = \frac{\phi^k \mathcal{N}\left(\mu_x^k, \Sigma^k\right)(x)}{\sum_{i=1}^{N} \phi^k \mathcal{N}\left(\mu_x^k, \Sigma^k\right)(x)}, \quad k = 1, \dots, N}$$

$$\boxed{\hat{y}(x,u) = \sum_{k=1}^{N} w^k(x)\left[\mu_y^k + A^k\left(x - \mu_x^k\right) + B^k u\right]}$$

3. Finally, we may choose which model to apply at x stochastically according to the above weights

$$\boxed{\hat{y}(x,u) = \mu_y^z + A^z(x - \mu_x^z) + B^z u, \quad P(z = k|x) = w^k(x)}$$

   This is easily accomplished by letting ξ be a random variable sampled from a uniform distribution over $[0,1]$. Then let

$$\boxed{z = k \quad if \quad \sum_{i=1}^{k-1} w^i(x) \le \xi < \sum_{i=1}^{k} w^i(x)}$$

   Choosing the linear models stochastically gives us a generative probabilistic model of the system's dynamics which may be useful in some cases. In such situations, the statistical properties of the dynamics are of greater interest than accurate prediction of the next state.

Implementation Overview

Form data matrices

$$X = [x_1 \quad \cdots \quad x_m] \in \mathbb{R}^{n\times m}$$

$$U = [u_1 \quad \cdots \quad u_m] \in \mathbb{R}^{n\times m}$$

$$Y = [y_1 \quad \cdots \quad y_m] \in \mathbb{R}^{n\times m}$$

Initialize variables

$$A^k = 0_{n\times n}, \quad B^k = 0_{n\times q}, \quad \mu_y^k = 0_n, \quad R^k = 0_{n\times n}, \quad \Sigma^k = \sigma_0 I_n, \quad \phi^k = \frac{1}{N}$$

Choose μ_x^k randomly in training data $k = 1, \ldots, N$

Initialize the joint probability for calculating the expectation

$$P_{y,x,u,z}(j,k) = \mathcal{N}\left(\mu_x^k, \Sigma^k\right)\left(X(:,j)\right)$$

Loop until convergence {

Expectation step:

$$\bar{Q}(j,k) = \frac{P_{y,x,u,z}(j,k)}{\sum_{i=1}^{N} P_{y,x,u,z}(j,i)}$$

$$\bar{Q} = P_{y,x,u,z} \oslash \left(P_{y,x,u,z}\mathbf{1}_{N\times N}\right)$$

where $\oslash$ is element-wise division.

Maximization step

Loop over $k = 1, \ldots, N$ models {

$$\mu_x^k = \frac{X\bar{Q}(:,k)}{\mathbf{1}_m^T\bar{Q}(:,k)}$$

Let $Z^k = X - \mu_x^k \mathbf{1}_m^T$

$$\Sigma^k = \frac{Z^k diag[\bar{Q}(:,k)]\left(Z^k\right)^T}{\mathbf{1}_m^T\bar{Q}(:,k)}$$

Let $\Xi^k = \begin{bmatrix} Z^k \\ U \\ \mathbf{1}_m^T \end{bmatrix}$

$$\tilde{A}^k = \begin{bmatrix} A^k & B^k & \mu_y^k \end{bmatrix} = Y\, diag[\bar{Q}(:,k)]\left(\Xi^k\right)^T\left(\Xi^k\, diag[\bar{Q}(:,k)]\left(\Xi^k\right)^T\right)^{-1}$$

```
Let $V^k = Y - \tilde{A}^k \Xi^k$

        $R^k = \frac{V^k \; diag[\bar{Q}(:,k)] \left(V^k\right)^T}{\mathbf{1}_m^T \bar{Q}(:,k)}$

        $\phi^k = \frac{1}{m} \mathbf{1}_m^T \bar{Q}(:,k)$

    Loop over $j = 1, \dots, m$ {

    Update joint probability for calculating the expectation

        $P_{y,x,u,z}(j,k) = \mathcal{N}\left(0_n, R^k\right) \left(V^k(:,j)\right) \mathcal{N}\left(\mu_x^k, \Sigma^k\right)\left(X(:,j)\right) \phi^k$

    } End loop over points $j$

} End loop over models $k$

} End EM convergence loop
```

Simple Illustration on 1D Function

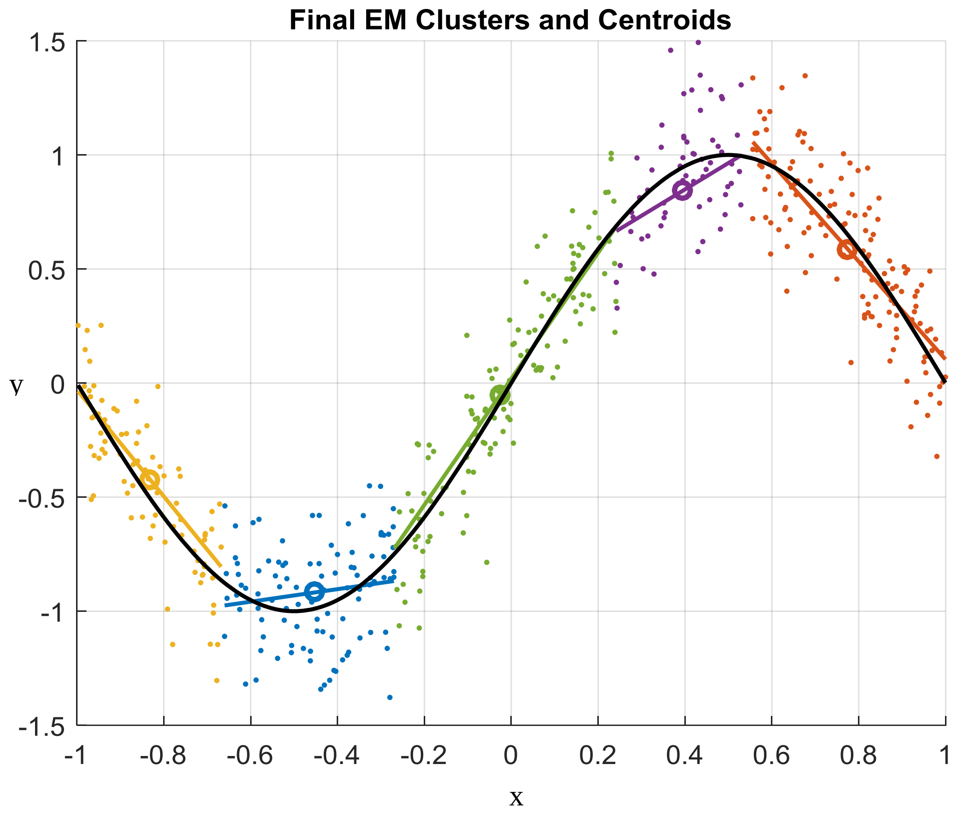
The function $\sin(\pi x)$ was sampled at 500 random points over $[-1,1]$ with Gaussian noise of standard deviation 0.2

$$y_j = \sin\left(\pi x_j\right) + v_j, \quad v_j \sim \mathcal{N}(\mu = 0, \sigma = 0.2)$$

Five linear models were fit to the data using the above EM algorithm. The following figure shows the linear models with their centroids. The points are colored according to maximum likelihood assignment to each model.

$$k^*\left(x_j\right) = \underset{k=1,\dots,N}{\operatorname{argmax}} \, \phi^k \mathcal{N}\left(\mu_x^k, \Sigma^k\right)\left(x_j\right)$$

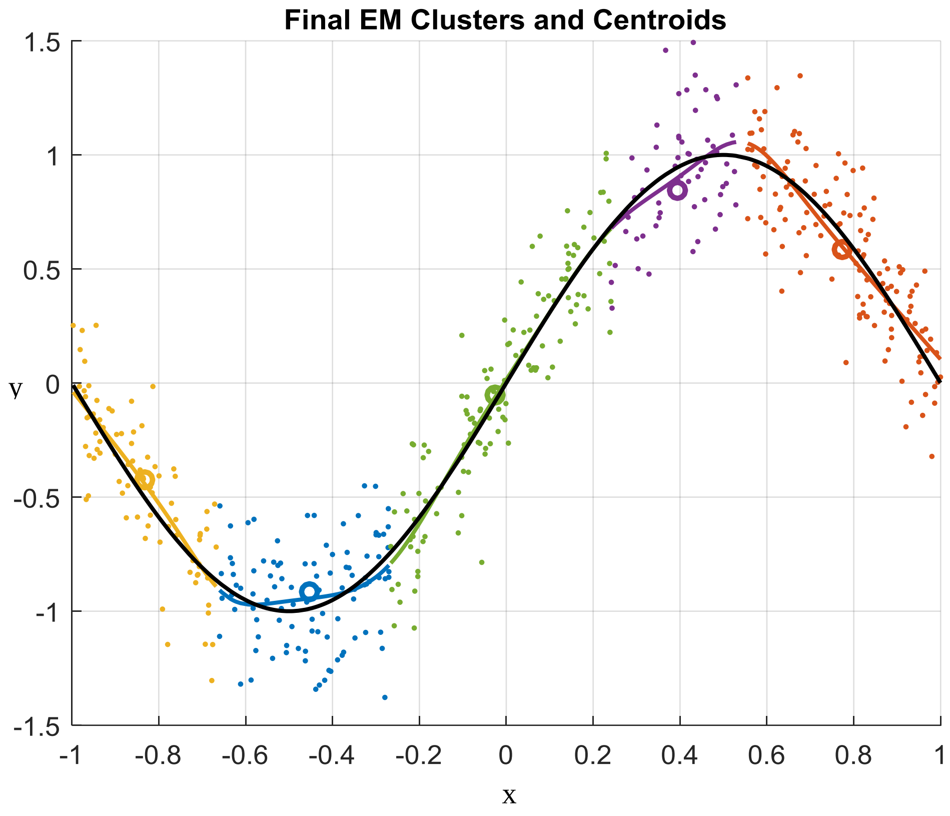
$$\hat{y}_j = \mu_y^{k^*} + A^{k^*}\left(x_j - \mu_x^{k^*}\right)$$



The following figure was produced by combining all the linear models at each point with weights given by the likelihood

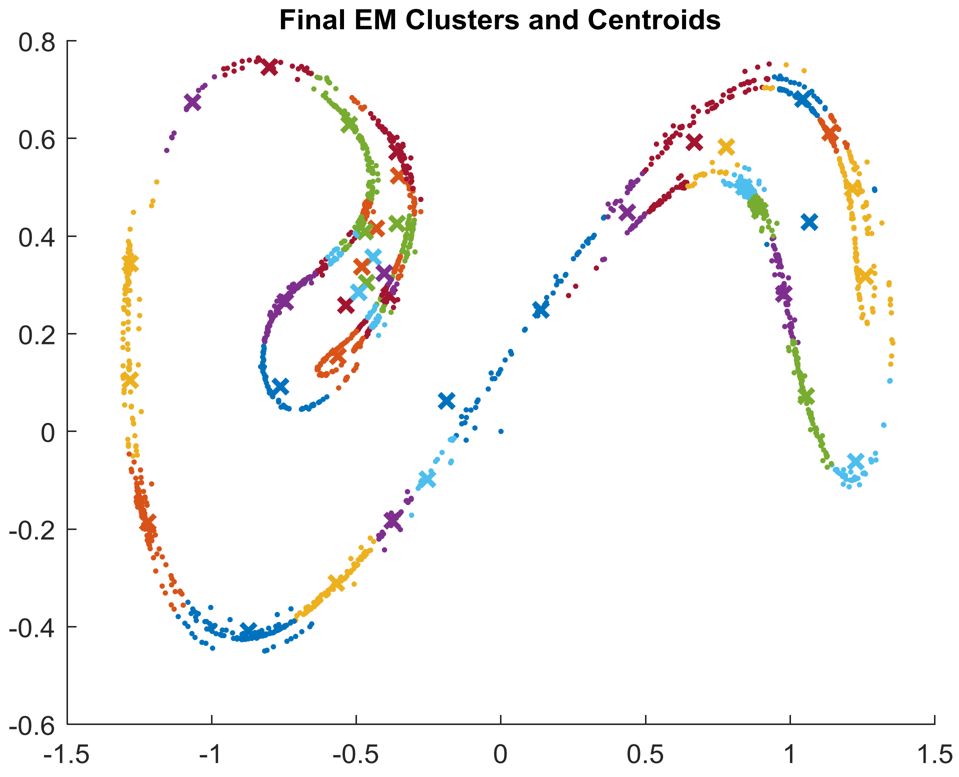
$$\hat{y}(x) = \frac{\sum_{k=1}^{N}\left[\phi^k \mathcal{N}\left(\mu_x^k, \Sigma^k\right)(x)\right]\left[\mu_y^k + A^k\left(x_j - \mu_x^k\right)\right]}{\sum_{k=1}^{N}\phi^k \mathcal{N}\left(\mu_x^k, \Sigma^k\right)(x)}$$

We observe that this gives a smooth approximation, whereas the above method is discontinuous.



Application to Poincare Map of Forced Duffing Equation

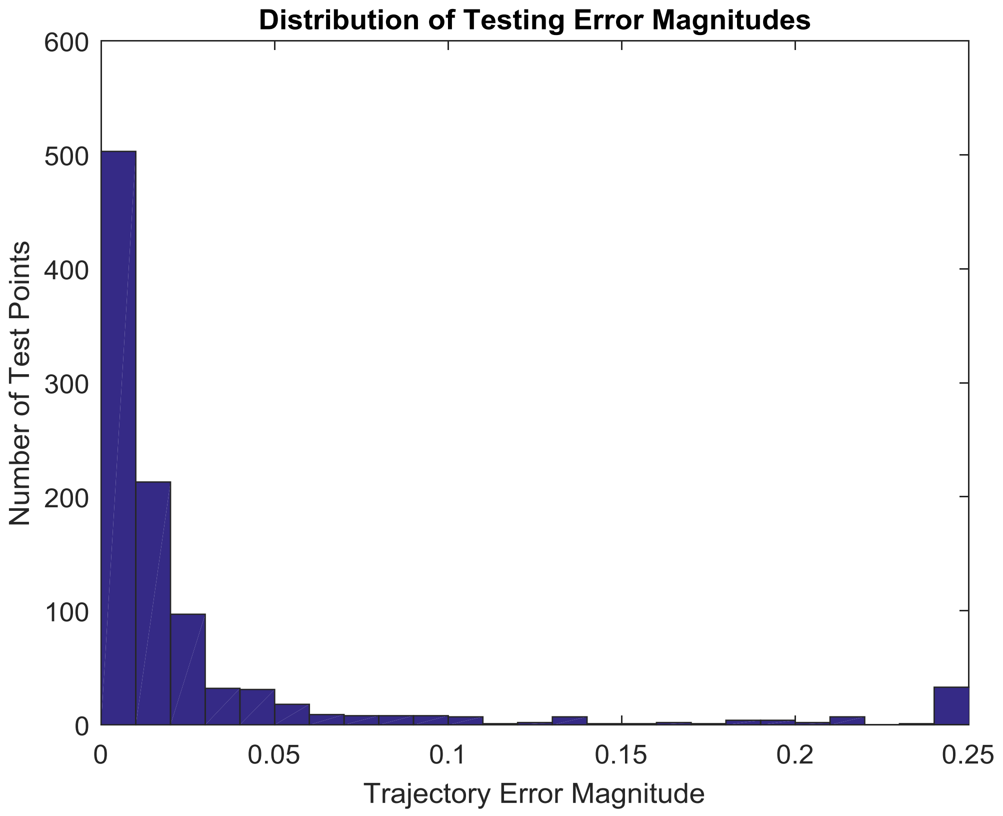
The initial centroids were chosen randomly in the training data. $m = 2000$ training points were used with randomly chosen linear combination of $N_b = 5$ Fourier basis forcing terms. The coefficients on these terms were taken to be the inputs u. The maximum coefficients on the forcing terms was set to be $\epsilon_p = 10^{-3}$. The EM algorithm was use to learn the parameters of $N = 40$ linear models. Convergence was determined when the maximum change in centroid locations was less than 10^{-4}. In order to avoid degenerate covariance matrices, the minimum singular values of Σ^k and R^k were constrained to be $\sigma_{min}(\Sigma^k) \geq \epsilon_\Sigma = 10^{-4}$ and $\sigma_{min}(\mathrm{R}^k) \geq \epsilon_\mathrm{R} = 10^{-4}$. The following figure shows the centroids of the linear models. The points of the Poincare map are colored according to maximum likelihood classification.



$m = 1000$ points were used for testing starting from a random initial condition and random forcing perturbations. The points were classified by maximum likelihood using the learned distributions

$$k^*(x_j) = \underset{k=1,\ldots,N}{\operatorname{argmax}} \phi^k \mathcal{N}(\mu_x^k, \Sigma^k)(x_j)$$

$$\hat{y}_j = \mu_y^{k^*} + A^{k^*}(x_j - \mu_x^{k^*}) + B^{k^*} u_j$$



$$MSE = \frac{1}{m}\sum_{j=1}^{m}\left\|y_j - \hat{y}_j\right\|^2$$

$$RMSE = \sqrt{\frac{1}{m}\sum_{j=1}^{m}\left\|y_j - \hat{y}_j\right\|^2}$$

$$Mean\ Dist.Error = \frac{1}{m}\sum_{j=1}^{m}\left\|y_j - \hat{y}_j\right\|$$

	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Average
MSE	0.0340	0.0212	0.0207	0.0138	0.0346	0.0249
RMSE	0.1843	0.1458	0.1437	0.1175	0.1861	0.1555
Mean Dist. Error	0.0514	0.0482	0.0399	0.0337	0.0565	0.0459
Median Dist. Error	0.0122	0.0124	0.0108	0.0099	0.0121	0.0115

Alternate Approach

Another approach is to successively fit collections of linear models and re-classify points. Gaussian densities are fit to the point clusters assigned to each linear model. These densities are used to perform weighted linear regression based on distance to each centroid.

```
Repeat until convergence {

    Loop over $k = 1, \dots, N$ linear models {

        1. Fit Gaussian density to training points $S^k = \{j_1^k, \dots, j_{m_k}^k\}$ assigned to model $k$

            $\mu_x^k = \frac{1}{m_k}\sum_{j\in S^k} x_j, \quad \Sigma^k = \frac{1}{m_k}\sum_{j\in S^k}(x_j - \mu_x^k)(x_j - \mu_x^k)^T$

        Calculate prior probability of model

            $\phi^k = \frac{m_k}{m}$

        2. Determine weights on each training point for use in linear regression

            $w_j = \mathcal{N}(\mu_x^k, \Sigma^k)(x_j), \quad W = \begin{bmatrix} w_1 & 0 & \\ 0 & \ddots & 0 \\ & 0 & w_m \end{bmatrix}$

        The algorithm can be sped up greatly by restricting the training points to be in $S^k$.

        3. Fit a linear models by solving the following weighted linear regression problem

            $\min_{\tilde{A}\in\mathbb{R}^{n\times(n+q+1)}} \left\| \sqrt{W}\left( Y^T - \underbrace{\begin{bmatrix} X^T - \mathbf{1}_m(\mu_x^k)^T & U^T & \mathbf{1}_m \end{bmatrix}}_{Z^T} \tilde{A}^T \right) \right\|_F^2$

            $\begin{bmatrix} A^k & B^k & \mu_y^k \end{bmatrix} = \tilde{A} = Y\sqrt{W}\left(Z\sqrt{W}\right)^+$

    }

    Loop over $j = 1, \dots, m$ training points{

        $S^k = \emptyset \quad k = 1, \dots, N$

        4. Assign a model to each training point according to minimum error

            $k_j^* = \operatorname*{argmin}_k \left\| y_j - \left[\mu_y^k + A^k(x_j - \mu_x^k) + B^k u_j\right] \right\|$

            $S^{k^*} := S^{k^*} \cup \{j\}$
        The argmin can also be taken over a smaller subset of models to avoid assigning a model
        whose centroid is far away. A successful method is to simply assign a training point to
        one of the two nearest linear models according to minimum error.
    }
} End convergence loop
```

New points are classified according to the maximum likelihood linear model

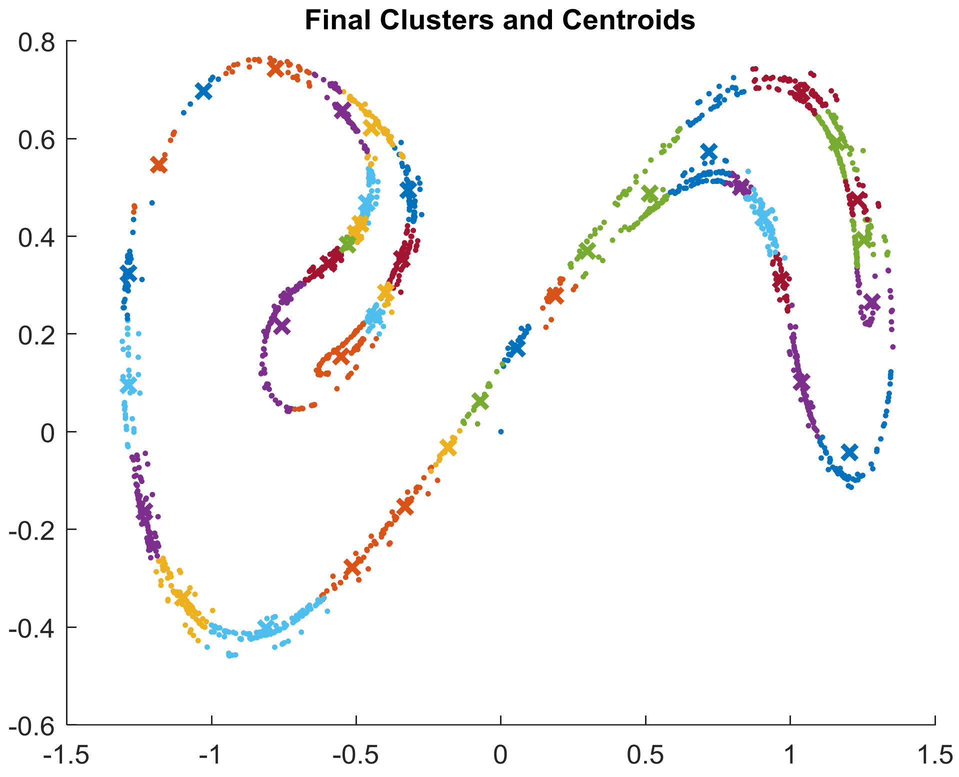
$$\boxed{k^*(x_j) = \underset{k=1,\ldots,N}{\operatorname{argmax}} \phi^k \mathcal{N}\left(\mu_x^k, \Sigma^k\right)(x_j)}$$

$$\boxed{\hat{y}_j = \mu_y^{k^*} + A^{k^*}\left(x_j - \mu_x^{k^*}\right) + B^{k^*} u_j}$$

It should be noted that the EM algorithm is guaranteed to converge since it produces a monotone increase in the likelihood function. The above alternative algorithm is not guaranteed to converge and has been shown to sometimes oscillate between two or more solutions. The method is more likely to oscillate when there are more than two models to choose from when assigning a given point.

Application to Poincare Map of Forced Duffing Equation

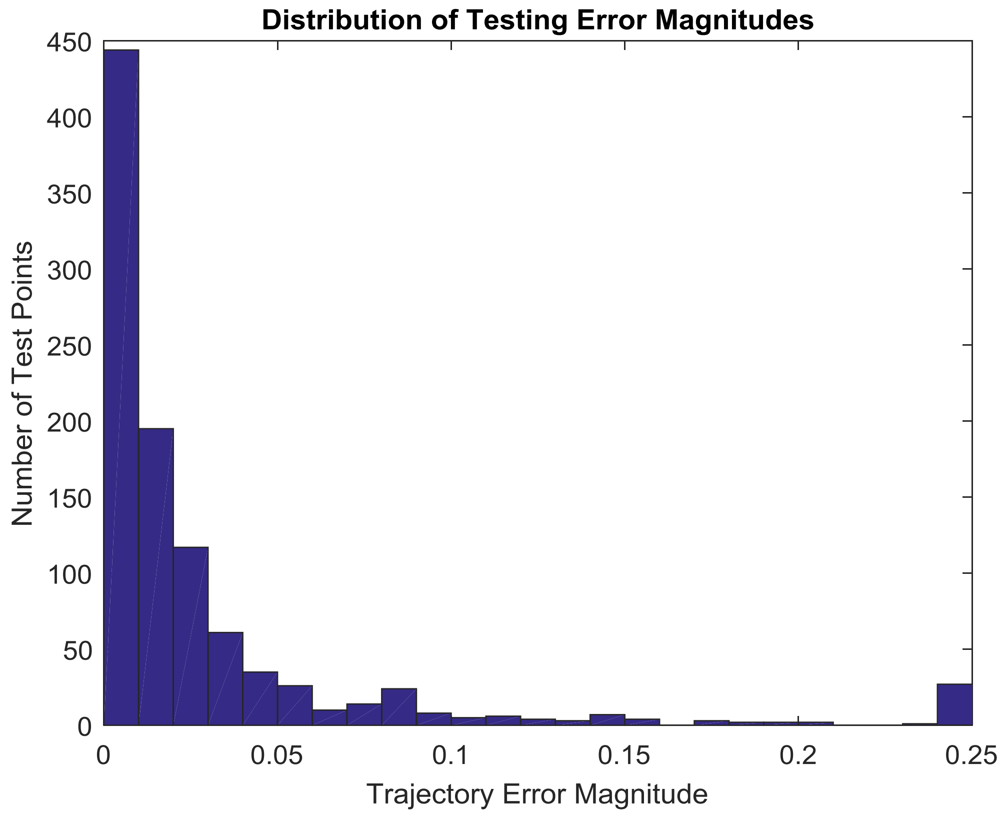
The initial centroids were chosen randomly in the training data. $m = 2000$ training points were used with randomly chosen linear combination of $N_b = 5$ Fourier basis forcing terms. The coefficients on these terms were taken to be the inputs u. The maximum coefficients on the forcing terms was set to be $\epsilon_p = 10^{-3}$. The EM algorithm was use to learn the parameters of $N = 40$ linear models. Convergence was determined when the maximum change in centroid locations was less than 10^{-10}. In order to avoid degenerate covariance matrices, the minimum singular values of Σ^k and R^k were constrained to be $\sigma_{min}(\Sigma^k) \geq \epsilon_\Sigma = 10^{-6}$. The following figure shows the centroids of the linear models. The points of the Poincare map are colored according to maximum likelihood classification. The algorithm was accelerated by only fitting the kth linear model to the points in S^k. Points were classified according to minimum error between the two nearest centroids.



$m = 1000$ points were used for testing starting from a random initial condition and random forcing perturbations. The points were classified by maximum likelihood using the learned distributions

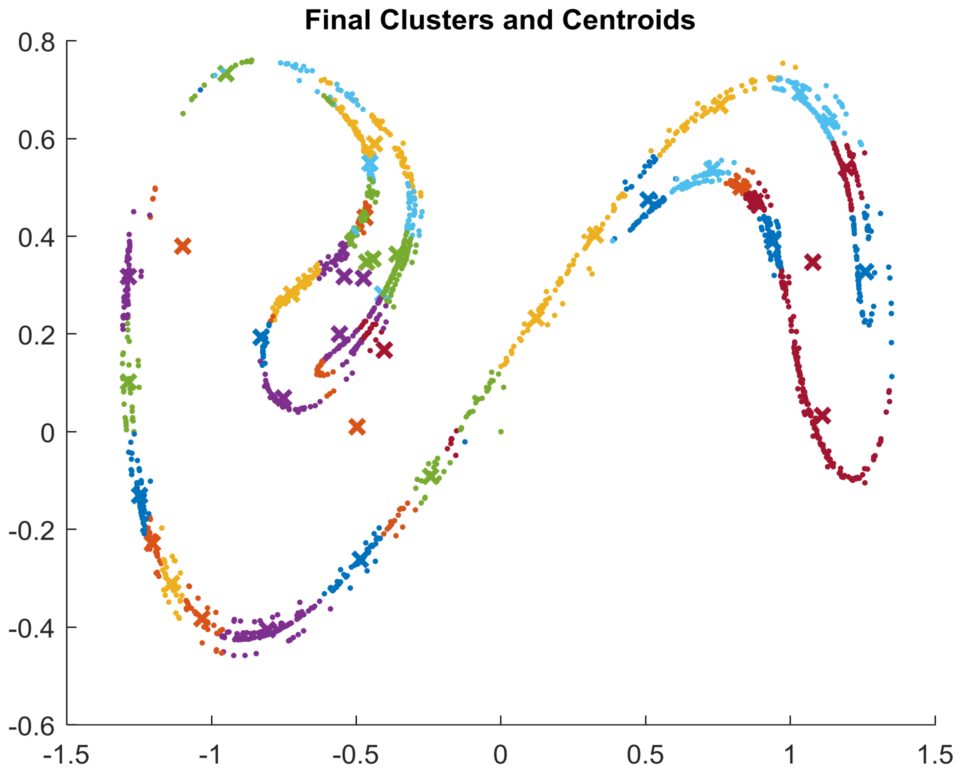
$$k^*(x_j) = \underset{k=1,\dots,N}{\operatorname{argmax}}\, \phi^k \mathcal{N}(\mu_x^k, \Sigma^k)(x_j)$$

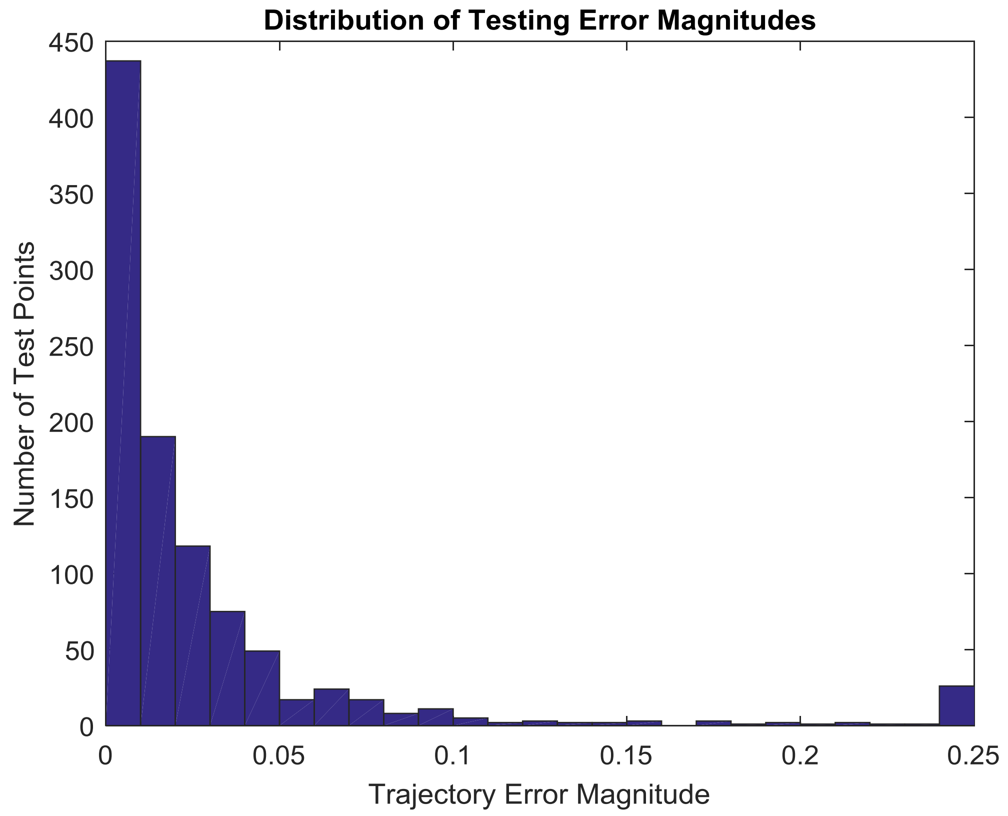
$$\hat{y}_j = \mu_y^{k^*} + A^{k^*}(x_j - \mu_x^{k^*}) + B^{k^*} u_j$$



	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Average
MSE	0.0189	0.4789	0.0448	0.0601	0.0164	0.1238
RMSE	0.1376	0.6920	0.2116	0.2451	0.1281	0.2829
Mean Dist. Error	0.0389	0.1070	0.0671	0.0485	0.0425	0.0608
Median Dist. Error	0.0121	0.0117	0.0131	0.0145	0.0132	0.0127

Next, the number of nearest linear models used for selection was increased to 10. We observe that cluster centroids were not as closely confined to training data lying on a curve. A greater number of training points lying near separate adjacent curves are assigned to the same model. The centroids are no longer confined to the curves.


Final Clusters and Centroids
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-1.5
-1
-0.5
0
0.5
1
1.5


Distribution of Testing Error Magnitudes
Number of Test Points
450
400
350
300
250
200
150
100
50
0
0
0.05
0.1
0.15
0.2
0.25
Trajectory Error Magnitude

Conclusion

- We have derived an EM algorithm which is shown to be effective for fitting a mixture of linear models to training data.
- Estimates $\hat{y}$ can be reconstructed at x, u from the linear models and their densities in a number of ways. Three possibilities are discussed:
    1. Use only the maximum likelihood linear model at x
    2. Combine several, or all linear models weighted by their likelihood at x
    3. Stochastically choose which linear model to apply at x with probabilities given by the likelihood of each model at x
- It is important to limit the minimum singular values in the learned covariance matrices to prevent degeneracies.
- The training data are approximately confined to curves in feature space for the forced Duffing example. We observe that the same linear model was applied to points near different adjacent curves. Consequently, the centroids are not confined to these curves.
- The alternative algorithm runs much more quickly than EM when regression is confined to points in S^k. However, it is not identical to EM and model errors are slightly higher. Also, convergence is not guaranteed for the alternative algorithm. The EM algorithm is guaranteed to converge since it produces a monotone increase in the likelihood function at each iteration.
- When more models are available for assigning points, the centroids depart from the curves on which the data lie. Further numerical experiments are needed to show how this affect the model error.
- The appropriate number of models as well as the other parameters can be determined using cross-validation.
- It is possible to fit higher order models simply by including nonlinear terms in x.
    - Locally weighted kernel regression could also be used to fit nonlinear models to point clusters.
    - Could be easier to fit many local models with regularized low-order nonlinearities than a single global model with high-order nonlinearity.
    - Model centroids become natural locations to study qualitative dynamics.
- Either algorithm could be applied to high-dimensional systems by first doing kernel PCA. This represents the dynamics of the system on a low-dimensional manifold. Then we can fit a mixture of linear models in the low-dimensional space.
- The M-step of the algorithm is easily parallelized, possibly allowing for accelerated calculation. Each model $k = 1, \ldots, N$ may be simultaneously updated.
- Suppose a collection of linear models with Gaussian densities has been fit to the data. We may determine regions where each model applies using the maximum likelihood criteria. These regions are good candidates for generalized rectangles of a Markov partition. We may look at how points transition between the different models in order to study the symbolic dynamics of the system.
    - I wonder if it is possible to use this approach to detect Smale horseshoes.
    - Transition network topology may be used to classify systems or to detect bifurcations. For example, we might look at the number of closed loops.

- The linear systems make it easy to approximate Lyapunov exponents.

Introducing Locally-Weighted Kernel Regression

The weighted linear regression step performed during the maximization phase of the above EM algorithm is extensible to fitting nonlinear models using a kernel method. Let us restrict our attention to approximating the lth component $y(l) = f_l(\xi)$, $\xi = [x^T \quad u^T \quad 1]^T$ using locally weighted regression on the training data

$$\Xi = \begin{bmatrix} x_1 & \cdots & x_m \\ u_1 & \cdots & u_m \\ 1 & \cdots & 1 \end{bmatrix} \in \mathbb{R}^{(n+q+1)\times m}, \quad \underline{y}(l) = \begin{bmatrix} y_1(l) \\ \vdots \\ y_m(l) \end{bmatrix} \in \mathbb{R}^m$$

In the proposed EM algorithm the data used in fitting the kth is weighted according to

$$w_j = \bar{Q}_j^k \Longrightarrow W \triangleq \begin{bmatrix} \bar{Q}_1^k & & \\ & \ddots & \\ & & \bar{Q}_m^k \end{bmatrix} \in \mathbb{R}^{m\times m}$$

We set up the weighted regression problem in a high-dimensional Reproducing Kernel Hilbert Space H by temporarily introducing a feature map from the data ξ into a high-dimensional Hilbert space $\Phi: \mathbb{R}^{n+q+1} \to H$. Then, we wish to find the combination θ_l of the features in H which best represents the function $\hat{y}(l) = \langle \Phi(\xi), \theta_l \rangle_H \approx f_l(\xi)$. We therefore state the regression problem for variable $y(l)$ as follows

$$\min_{\theta_l \in H} \left\| \sqrt{W} \left(\underline{y}(l) - \langle \Phi(\Xi), \theta_l \rangle_H \right) \right\|_2^2 + \lambda \langle \theta_l, \theta_l \rangle_H$$

The parameter λ is used for regularization in order to minimize over-fitting of the nonlinear model. In the above equation, we define

$$\langle \Phi(\Xi), \theta_l \rangle \triangleq \begin{bmatrix} \langle \Phi(\xi_1), \theta_l \rangle \\ \vdots \\ \langle \Phi(\xi_m), \theta_l \rangle \end{bmatrix} \in \mathbb{R}^m, \quad \begin{matrix} \Phi(\Xi): \mathbb{R}^m \to H \\ \Phi(\Xi)v = v_1\Phi(\xi_1) + \cdots + v_m\Phi(\xi_m) \end{matrix}$$

We observe that θ_l must be in the range of $\Phi(\Xi)$ in H. Otherwise we could write $\theta_l = \hat{\theta}_l + \tilde{\theta}_l$ with $\hat{\theta}_l \in \mathcal{R}(\Phi(\Xi))$ and $\hat{\theta}_l \in \mathcal{R}(\Phi(\Xi))^{\perp}$. Then $\langle \Phi(\Xi), \theta_l \rangle_H = \langle \Phi(\Xi), \hat{\theta}_l \rangle_H + \langle \Phi(\Xi), \tilde{\theta}_l \rangle_H = \langle \Phi(\Xi), \hat{\theta}_l \rangle_H$. But $\langle \theta_l, \theta_l \rangle_H = \langle \hat{\theta}_l, \hat{\theta}_l \rangle_H + \langle \tilde{\theta}_l, \tilde{\theta}_l \rangle_H \geq \langle \hat{\theta}_l, \hat{\theta}_l \rangle_H$. Thus, we must have $\tilde{\theta}_l = 0$. We therefore represent the solution $\theta_l = \hat{\theta}_l = \Phi(\Xi)\tilde{a}_l$ as a linear combination of the m vectors $\Phi(\xi_j) \in H$. Thus, the regression problem becomes finite-dimensional

$$\min_{\tilde{a}_l \in \mathbb{R}^m} \left\| \sqrt{W} \left(\underline{y}(l) - \langle \Phi(\Xi), \Phi(\Xi) \rangle_H \tilde{a}_l \right) \right\|_2^2 + \lambda \tilde{a}_l^T \langle \Phi(\Xi), \Phi(\Xi) \rangle_H \tilde{a}_l$$

Let

$$K \triangleq \langle \Phi(\Xi), \Phi(\Xi)\rangle_H = \begin{bmatrix} \langle \Phi(\xi_1), \Phi(\xi_1)\rangle_H & \langle \Phi(\xi_1), \Phi(\xi_2)\rangle_H & \cdots & \langle \Phi(\xi_1), \Phi(\xi_m)\rangle_H \\ \langle \Phi(\xi_2), \Phi(\xi_1)\rangle_H & \langle \Phi(\xi_2), \Phi(\xi_2)\rangle_H & \cdots & \langle \Phi(\xi_2), \Phi(\xi_m)\rangle_H \\ \vdots & \vdots & \ddots & \vdots \\ \langle \Phi(\xi_m), \Phi(\xi_1)\rangle_H & \langle \Phi(\xi_m), \Phi(\xi_2)\rangle_H & \cdots & \langle \Phi(\xi_m), \Phi(\xi_m)\rangle_H \end{bmatrix} \in \mathbb{R}^{m\times m}$$

Clearly, the matrix K is symmetric and can be assumed invertible. Otherwise, we can simply project onto the orthogonal basis of eigenvectors corresponding to nonzero eigenvalues of K and form the pseudoinverse instead. We now proceed to solve the resulting standard ridge regression problem

$$\min_{\tilde{a}_l\in\mathbb{R}^m} \left\|\sqrt{W}\left(\underline{y}(l) - K\tilde{a}_l\right)\right\|_2^2 + \left\|\sqrt{\lambda K}\tilde{a}_l\right\|_2^2 = \min_{\tilde{a}_l\in\mathbb{R}^m} \left\|\begin{bmatrix}\sqrt{W}y_l \\ 0\end{bmatrix} - \begin{bmatrix}\sqrt{W}K \\ \sqrt{\lambda K}\end{bmatrix}\tilde{a}_l\right\|_2^2$$

which is reduced to standard linear regression with normal equations

$$[K\sqrt{W} \quad \sqrt{\lambda K}]\begin{bmatrix}\sqrt{W}K \\ \sqrt{\lambda K}\end{bmatrix}\tilde{a}_l = [K\sqrt{W} \quad \sqrt{\lambda K}]\begin{bmatrix}\sqrt{W}\underline{y}(l) \\ 0\end{bmatrix}$$

$$(KWK + \lambda K)\tilde{a}_l = KW\underline{y}(l) \Longrightarrow \tilde{a}_l = (WK + \lambda I_m)^{-1}W\underline{y}(l)$$

Forming the estimator

$$\hat{y}(l) = \langle \Phi(\xi), \theta_l\rangle_H = \langle \Phi(\xi), \Phi(\Xi)\rangle_H \tilde{a}_l = \langle \Phi(\xi), \Phi(\Xi)\rangle_H (WK + \lambda I_m)^{-1}W\underline{y}(l)$$

Form the data matrix

$$Y = \begin{bmatrix} y_1(1) & \cdots & y_m(1) \\ \vdots & & \vdots \\ y_1(n) & \cdots & y_m(n)\end{bmatrix} = [y_1 \quad \cdots \quad y_m] \in \mathbb{R}^{n\times m}$$

and form the estimator for all components $l = 1, \dots, n$ simultaneously

$$\hat{y}^T = \langle \Phi(\xi), \Phi(\Xi)\rangle_H (WK + \lambda I_m)^{-1}WY^T$$

$$\hat{y} = \hat{f}(\xi) = YW(KW + \lambda I_m)^{-1}\langle \Phi(\Xi), \Phi(\xi)\rangle_H$$

If we introduce a kernel function $k(\xi_1, \xi_2) \triangleq \langle \Phi(\xi_1), \Phi(\xi_2)\rangle_H$ and $k(\xi) \triangleq \langle \Phi(\Xi), \Phi(\xi)\rangle_H = [\langle \Phi(\xi_1), \Phi(\xi)\rangle_H, \dots, \langle \Phi(\xi_m), \Phi(\xi)\rangle_H]^T$ then we have

$$\boxed{\hat{y} = \hat{f}(\xi) = YW(KW + \lambda I_m)^{-1}k(\xi), \quad [K]_{i,j} = \langle \Phi(\xi_i), \Phi(\xi_j)\rangle_H}$$

As an aside, the above equation is very easy to linearize if we know the derivative of the kernel with respect to one of its components $D_\xi k(\xi_0, \xi)$. We now can find the linearized Poincare map around ξ^*

$$\boxed{D_\xi \hat{f}(\xi^*)\Delta\xi = YW(KW + \lambda I_m)^{-1}\begin{bmatrix} D_\xi k(\xi_1, \xi^*) \\ \vdots \\ D_\xi k(\xi_m, \xi^*)\end{bmatrix}\Delta\xi = [A^k \quad B^k \quad *]\begin{bmatrix} x \\ u \\ 0\end{bmatrix}}$$

Higher order terms are obtained in the same way by differentiating the kernel. This quick and analytical linearization technique will likely be useful if the learned nonlinear models are to be used for estimation and/or control.

Finally, it should also be noted that it can be quite expensive to invert $KW + \lambda I_m$. Therefore, we can break the matrix into two parts by choosing only the largest $M < m$ weights $w_1, \dots, w_M$ to be nonzero. We form the sub-matrices

$$\tilde{K} = \begin{bmatrix} K_{11} & \cdots & K_{1M} \\ \vdots & & \vdots \\ K_{M1} & \cdots & K_{MM} \end{bmatrix}, \quad \tilde{W} = \begin{bmatrix} \bar{Q}_1^k & & \\ & \ddots & \\ & & \bar{Q}_M^k \end{bmatrix}, \quad \tilde{\Xi} = [\xi_1 \quad \cdots \quad \xi_M], \quad \tilde{Y} = [y_1 \quad \cdots \quad y_M]$$

The Schur complement can be used to show that the new estimator with only M nonzero weights is given by the same equation using the sub-matrices. With the intention of doing local modeling, this is a sensible thing to do in order to speed up the algorithm.

$$\boxed{\hat{y} = \hat{f}(\xi) = \tilde{Y}\tilde{W}\left(\tilde{K}\tilde{W} + \lambda I_M\right)^{-1}\tilde{k}(\xi)}$$

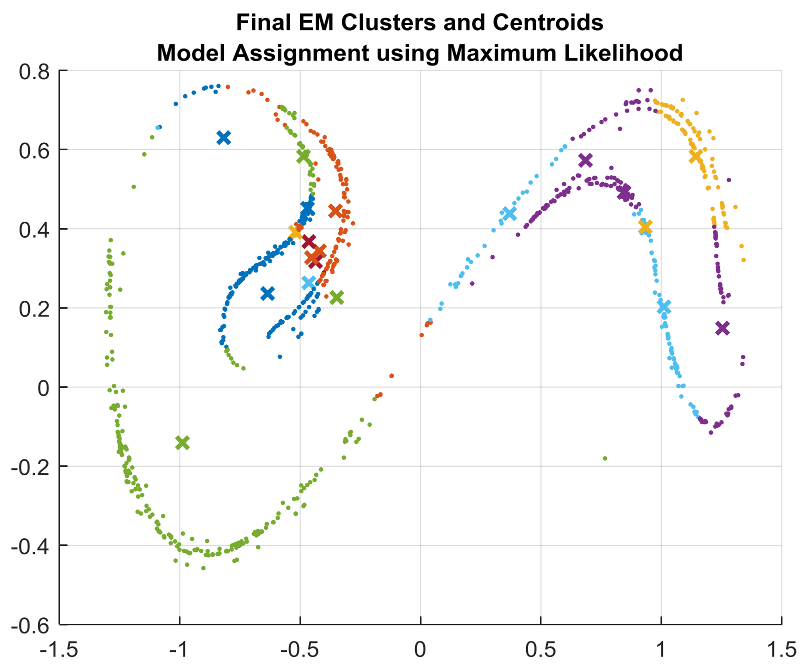
$$\tilde{k}(\xi) \triangleq \langle \Phi(\tilde{\Xi}), \Phi(\xi) \rangle_H = [\langle \Phi(\xi_1), \Phi(\xi) \rangle_H, \dots, \langle \Phi(\xi_M), \Phi(\xi) \rangle_H]^T$$

Application to Poincare Map of Forced Duffing Equation

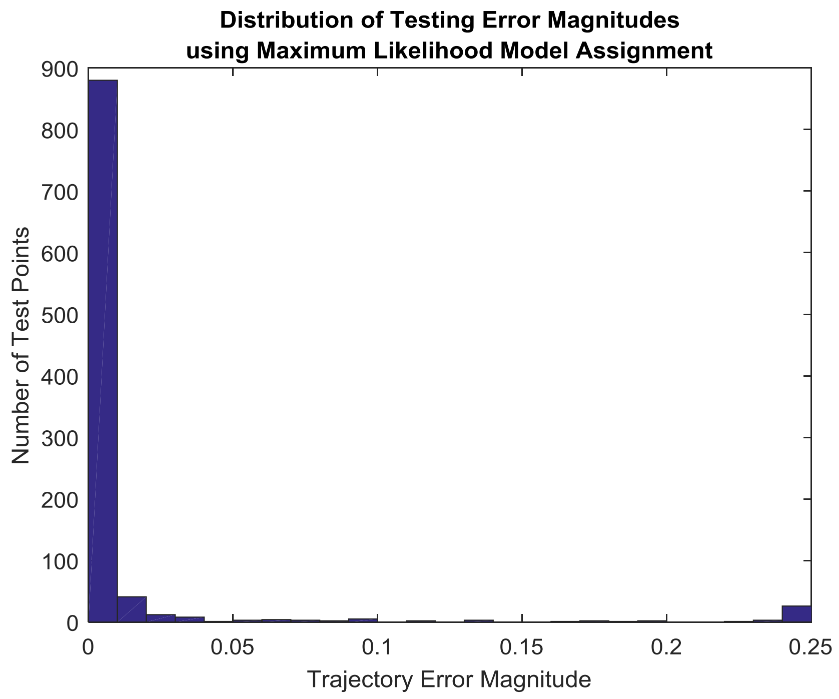
$N = 20$ nonlinear models were fit to $m = 2000$ training points generated by the forced Duffing equation with the same parameters and forcing as in the above experiments. The radial basis function was used as the kernel

$$k(\xi_1, \xi_2) = \exp\left(-\frac{\|\xi_1 - \xi_2\|_2^2}{2\sigma^2}\right)$$

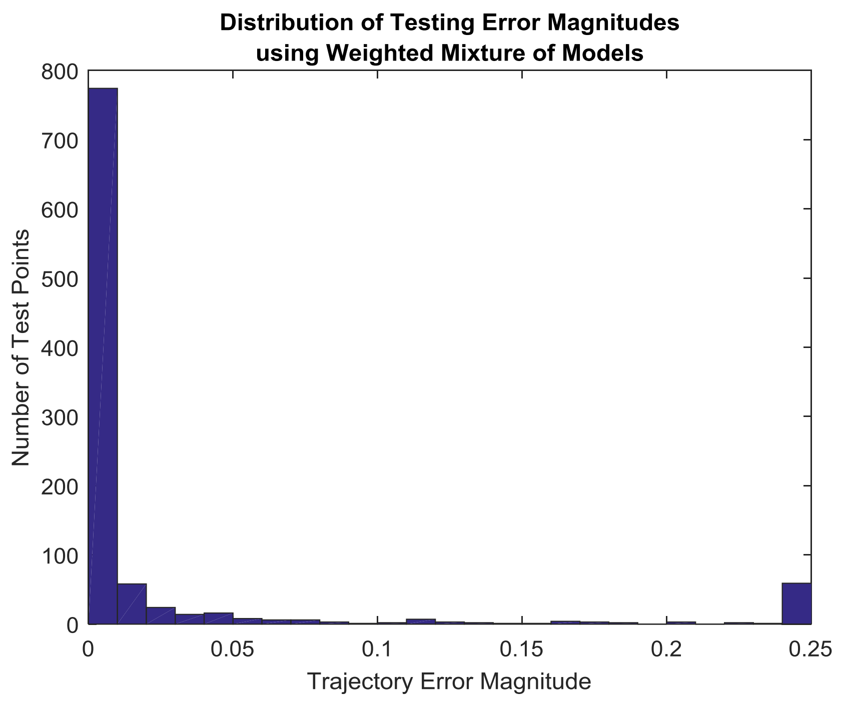
The subsets of the data used to train the kernel regression models were limited to $M = 500$ points. Cross-validation was used to select regularization parameters $\sigma = 0.1$ and $\lambda = 5 * 10^{-4}$. Iteration was stopped when the maximum change in the model centroids was less than 10^{-4}. Non-degeneracy was enforced using minimum singular values $\epsilon_\Sigma = 10^{-4}$, $\epsilon_\mathrm{R} = 10^{-4}$.



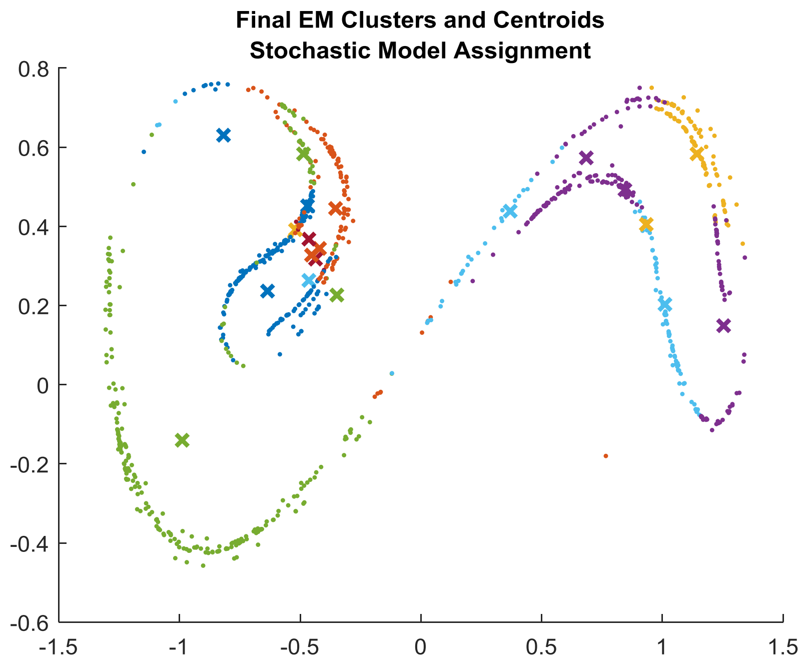
Final EM Clusters and Centroids
Model Assignment using Maximum Likelihood
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-1.5
-1
-0.5
0
0.5
1
1.5



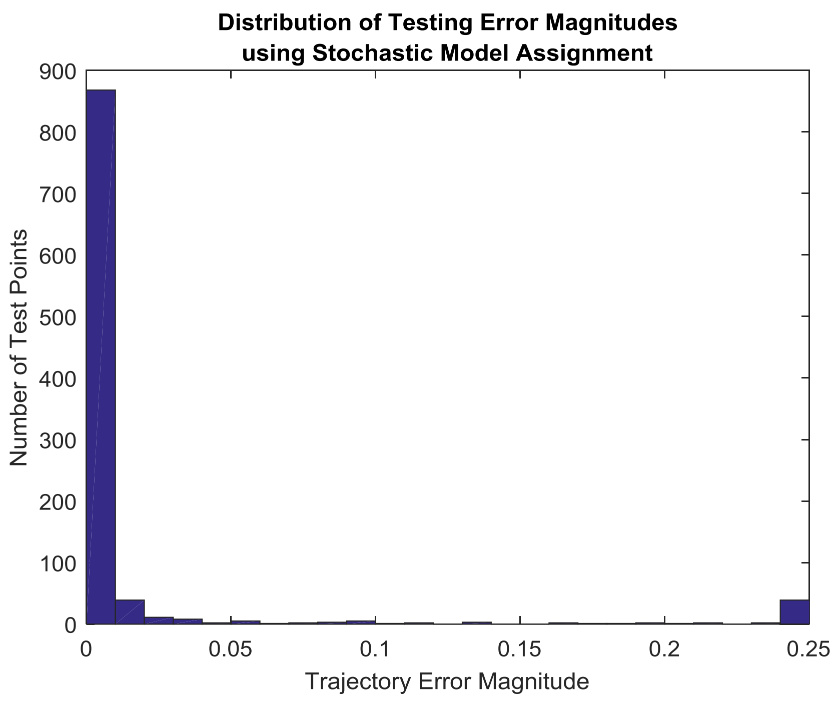
Distribution of Testing Error Magnitudes
using Maximum Likelihood Model Assignment
Number of Test Points
900
800
700
600
500
400
300
200
100
0
0
0.05
0.1
0.15
0.2
0.25
Trajectory Error Magnitude



Distribution of Testing Error Magnitudes
using Weighted Mixture of Models
Number of Test Points
Trajectory Error Magnitude
0
100
200
300
400
500
600
700
800
0
0.05
0.1
0.15
0.2
0.25

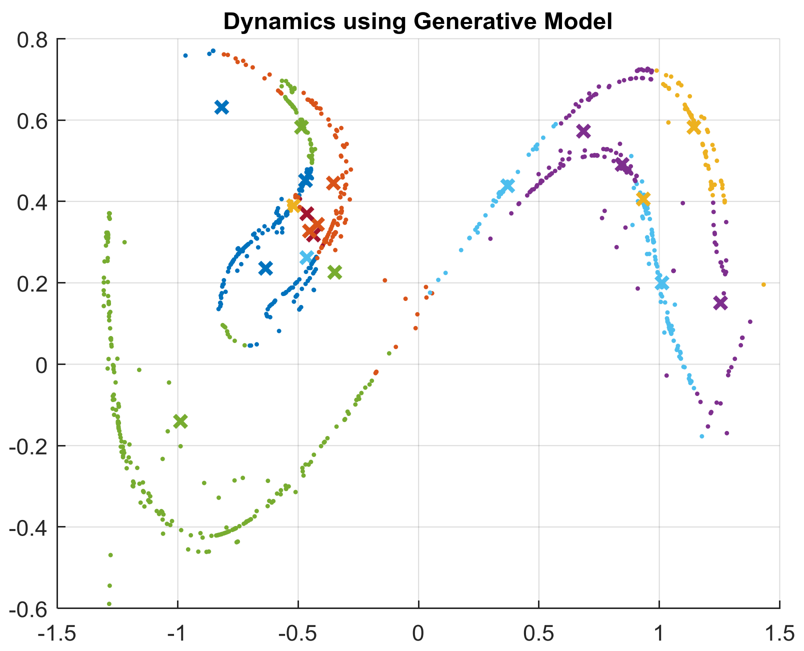


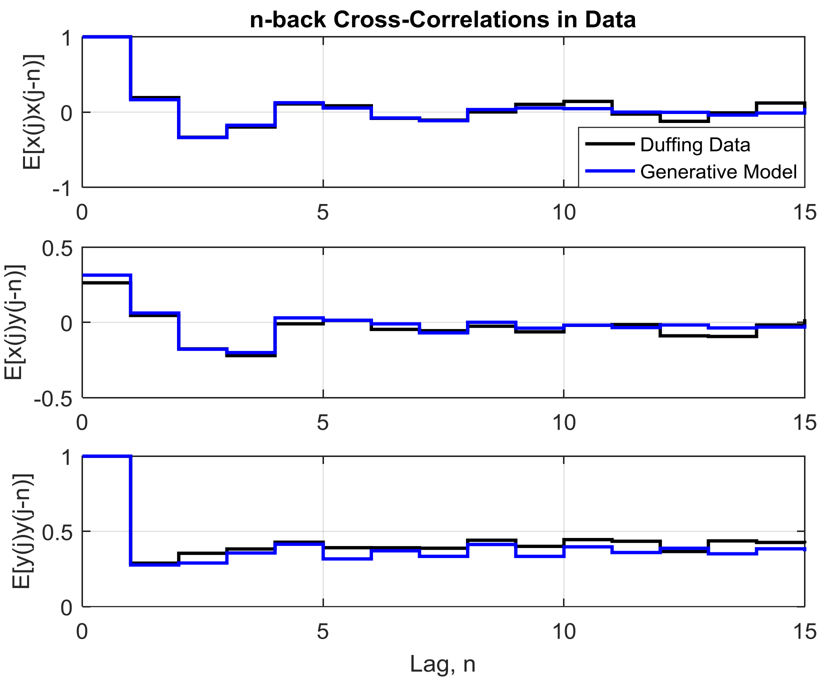
Final EM Clusters and Centroids
Stochastic Model Assignment
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
-1.5
-1
-0.5
0
0.5
1
1.5



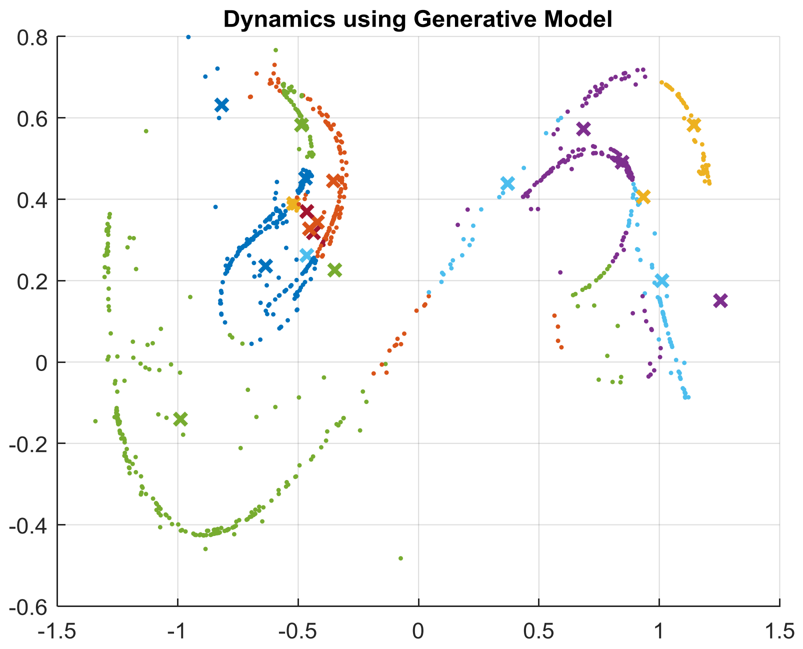
Demonstration as a Generative Model

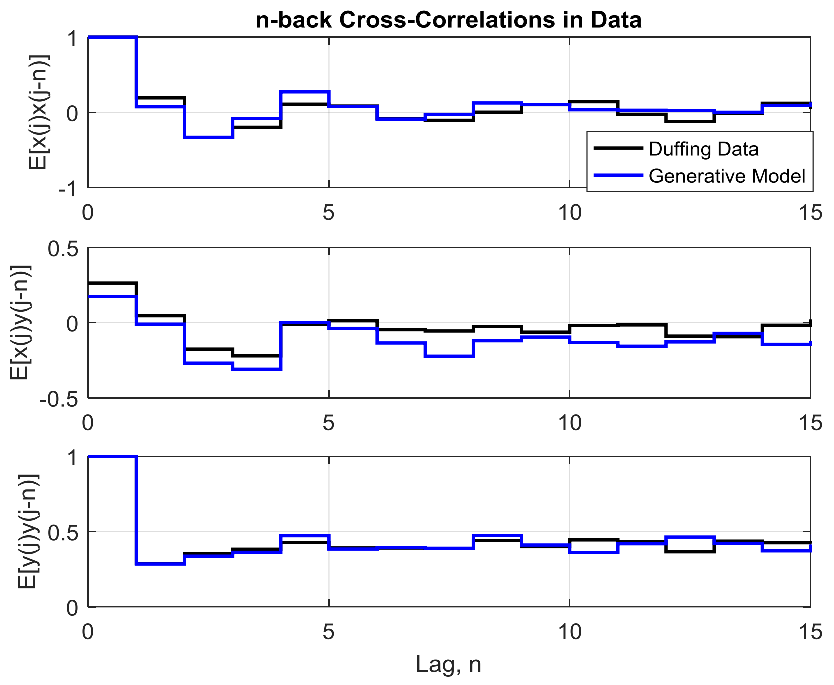
As an experiment, I decided to try using the leaned model to generate a Poincare map starting from the random initial condition $x_{IC} = [0.5902 \quad 0.5247]^T$ using zero forcing inputs. Maximum likelihood model assignment was used. The results appear very similar to the original Poincare map! I then investigated the replication of the Poincare map's statistical properties. Assuming the process is stationary; I looked at then n-step back covariances.



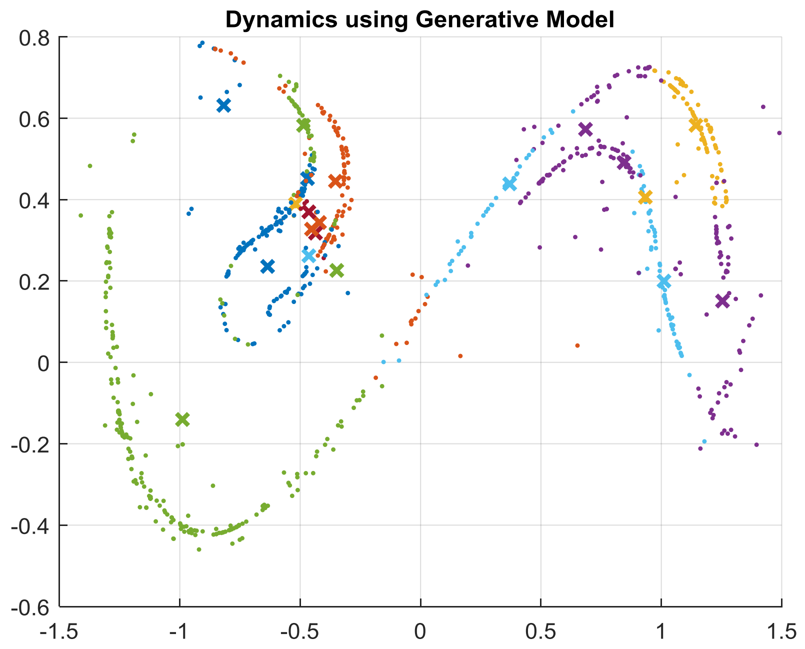


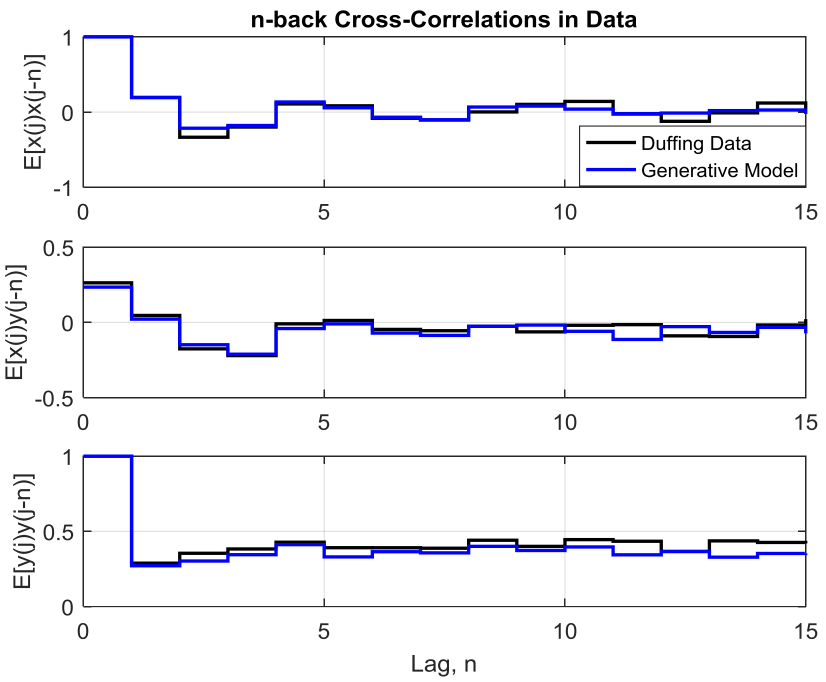
We see that the generative model using maximum likelihood assignment of nonlinear models does a good job capturing the n-back covariances in the Poincare map. Next, we try using the weighted mixture of model at each point





This does not seem to do as well as maximum likelihood assignment. This may be because each nonlinear model was only trained on a subset of the data and therefore might produce large, spurious estimates far away from its centroid. The exponential form of the RBF kernel may allow the estimates to grow more quickly at infinity than the Gaussian weights. Thus, the spurious estimates may be non-negligible even though they are given small weights. Finally I tried using stochastic model assignment, with nice-looking results




n-back Cross-Correlations in Data
E[x(j)x(j-n)]
E[x(j)y(j-n)]
E[y(j)y(j-n)]
Duffing Data
Generative Model
Lag, n
1
0
-1
0.5
-0.5
0
5
10
15