Course: Modelling of Measurement
Homework: Models from Data

Yidan Hu
Energy Department, Politecnico di Milano
Piazza Leonardo da Vinci, 32 20133 Milano, Italy
Email: yidan.hu@polimi.it

Abstract—**To describe the characteristics of nonlinear-dynamic systems, we can do it by measuring and obtaining observational data. Many physical laws have not yet been explored in mathematical models, and it is particularly important to build models from data that can be used to describe or even predict. When the dynamical system has non-linearity and time-variant characteristics, the conventional regression methods may not predict the model very well. The main focus of this course is on characterizing complex systems through dimensional reduction, sparse sampling, and dynamical systems modeling by many algorithms, such as Singular Value Decomposition, Dynamic Mode Decomposition, Sparse Identification of Nonlinear Dynamics, Koopman Operator Theory, Delay Coordinates and the Proper Orthogonal decomposition. Besides all the mentioned algorithms, machine learning and data analysis are explored in many aspects, such as regression and model selection, clustering and classification and neural network and deep learning. This article attempts to carry out a series of algorithmic programming based on the given data, all the code can be obtained on Github at https://github.com/yidan-hu/Homework-of-Yidan-Hu.git.**

1. Introduction and Overview

In this section, a brief and basic introduction of the course is presented. The goal is to build a model from a bunch of data which could be from simulation or measurement. How to find out a reduced space model which can represent the system and in order to predict the future is a key goal in this course. In the introduction section, main parts in the lecture are briefly presented and explained, and details could be found in the book [1]. Four parts of this book are briefly reviewed and the main ideas from lecture are explained in this sections.

Besides, many key themes are discussed in this course. First, how to find the dominant low-dimensional patterns of complex systems is sometimes the first step to do with the data. This low-dimensional patterns could enables efficient sensing and compact representations for modeling and control. The proper orthogonal decomposition (POD) including SVD is the main tool of dimension reduction. This pattern extraction has relationship with the second theme of how to find the proper coordinate to represent the model simply. Many techniques in the coordinate transforms are familiar to engineering, such as, spectral decompositions, the Fourier transform, generalized functions, etc. In this book, many data-driven coordinate transforms are explained, like, DMD, SINDy, Koopman, etc. The third theme is dynamical systems and control. When handling big data, first to apply the coordinate transforms to find approximated low-dimensional model and then to train a Neural Network (NN) based on this model is a kind of good method. The fourth themes is data-driven applied optimization, such as, finding optimal low-dimensional patterns, optimal sensor placement, machine learning optimization and optimal control. In class, the case about optimization of sensor placement was explained. In the following subsections, these four chapters are briefly summarized for future reference, some of key points are also explained.

1.1. Dimensionality Reduction and Transforms

SVD, Fourier and wavelet transforms, and sparsity and compressed sensing are described in this chapter. Any signal could be modeled by modes and the values on that modes. Fourier and wavelet are two types of coordinates transformation which utilize sinusoidal and and wavelet as basis modes. The desired coordinate systems are simplify, decouple and amenable to computation and analysis.

Besides, the SVD is a generalized concept of the fast Fourier transform (FFT). The SVD provides a numerically stable matrix decomposition that can be used for a variety of purposes and is guaranteed to exist. It could be used to obtain low-rank approximations to matrices and to perform pseudo-inverses of non-square matrices to find the solution of a system of equations Ax=b. Some of large matrices of complex systems actually contains few dominant patterns that explain the high-dimensional data. This SVD is a numerically robust and efficient method of extracting these patterns from data. Therefore, SVD is usually the first step when analysing data of dynamic systems in some algorithems like DMD, POD, classification methods, etc.

1.2. Machine Learning and Data Analysis

Machine learning is based on optimization techniques for data, and there are two broad categories: supervised machine learning and unsupervised machine learning. The goal is to find both low-rank subspace for optimally embedding the data, as well as regression methods for clustering and classification of different data types. Data mining, i.e. a principle set of mathematical methods for extracting meaningful features from data, as well as binning the data into distinct and meaningful patterns that can be exploited for decision making. All of this machine learning architecture is finding low-rank feature spaces that are informative and interpretable. The goal is to construct and exploit the intrinsic low-rank feature space of a given data set.

1. Supervised machine learning

Supervised learning means that machine learning data is labeled, and these labels can include data categories, data attributes, and feature point locations. These flags are used as expected effects to continually revise the machine's predictions. The specific first process is: train the machine through a large amount of labeled data, and the machine compares the predicted results with the expected results; then modifies the parameters in the model according to the comparison results, and outputs the predicted results again; then compare the predicted results with the expected results. The expected results are compared, repeated many times until convergence, and finally a model with a certain robustness is generated to achieve the ability of intelligent decision-making. Common supervised learning methods include classification and regression. Classification is to classify some instance data into appropriate categories, and its prediction results are discrete. Regression is to put the data on a "line", that is, to produce a fitted curve for discrete data, so its prediction results are continuous. The main algorithms used to perform regression and classification are Naive Bayes algorithm, support vector machine, decision tree and other algorithms.

2. Unsupervised machine learning

Unsupervised learning means that the data for machine learning is unlabeled. Machines explore and infer potential connections from unlabeled data. Common unsupervised learning includes clustering and dimensionality reduction. In the clustering work, since the data category is not known in advance, the distribution of the data samples in the feature space can only be analyzed, such as based on density or based on statistical probability models, etc. Similar data are grouped together. Dimensionality reduction is to reduce the dimensionality of data. Since the data itself has a huge amount and various attributes, if all data information is analyzed, it will increase the burden of training and storage space. Therefore, other methods such as principal component analysis can be used to balance the accuracy and efficiency by considering the main influencing factors and discarding the secondary factors. The book introduces K-means clustering, Hierarchical Clustering: Dendrogram, and Mixture Models and the Expectation-Maximization Algorithm.

3. Deep learning

A more typical and advanced machine learning framework evolved from Artificial Neural Network (ANN) is called deep learning. An artificial neural network consists of an input layer, a hidden layer and an output layer, and each layer consists of multiple neuron nodes. The calculation process of the neuron is as follows: each value in the input signal is multiplied by a weight, indicating the contribution of the signal to the result; then all products are summed; the summation result plus the bias parameter is activated by the activation function to obtain the neuron meta output. Neural Networks (NN) can be regarded as a map from input to output. The NN is built on trained data and validated with withdraw data, and used to predict with new data. Since the NNs have significant potential for overfitting of data, the cross -validation must be considered. NNs offer an amazingly flexible architecture for performing a diverse set of mathematical tasks, for instance, it can be used for future state predictions of dynamical systems.

1.3. Dynamic and control

Real-world systems are often nonlinear, dynamic, multi-scale, high-dimensional in space and time, with major underlying patterns that should be characterized and modeled for the ultimate goals of sensing, prediction, estimation, and control. Dynamical systems involve the analysis, prediction, and understanding of the behavior of differential equations or iteratively mapped systems that describe the evolution of the state of the system. The most pressing scientific and engineering problems of modern times do not apply to empirical models or derivations based on first principles. Dynamical systems provide a mathematical framework to describe the world around us, modeling the rich interactions between quantities that co-evolve over time. Characterize complex systems through dimensionality reduction, sparse sampling, and dynamic system modeling. Discover dynamics from data and find data-driven representations to make nonlinear systems suitable for linear analysis.

1.4. Reduced Order Models

Proper Orthogonal Decomposition (POD), an SVD algorithm applied to partial differential equations (PDE), is one of the most important dimensionality reduction techniques for studying complex spatiotemporal systems. Such systems are often exemplified by nonlinear partial differential equations that specify the evolution in time and space of a quantity of interest in a given physical, engineering, and/or biological system. Reduced-Order Models (ROMs) utilize POD patterns to project PDE dynamics into low-rank subspaces where simulations governing PDE models can be more easily evaluated to facilitate system understanding and computational efficiency. Importantly, the low-order models generated by ROM can significantly increase computational speed, potentially enabling expensive Monte Carlo simulations of PDE systems, optimization of parameterized PDE systems, and/or real-time control of PDE-based systems.

2. Theoretical Background

2.1. Singular Value Decomposition (SVD)

The SVD is one of the most popular dimensionality reduction methods, and it has been applied in a wide range of applications, including genomics, physics, and image processing. SVD is the underlying algorithm for principal component analysis (PCA).

Given the data matrix $X \in \mathbb{R}^{m\times n}$, the SVD decomposes X into the product of three matrices,

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T \tag{1}$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{m\times m}$ are unitary matrices, and $\Sigma \in \mathbb{R}^{m\times n}$ is a diagonal matrix with nonnegative entries. We denote the *i*th columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ as $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$, respectively. The diagonal elements of Σ, σ_i, are known as the singular values of $\boldsymbol{X}$, and they are written in descending order.

The rank of the data is defined to be *R*, which equals the number of nonzero singular values. Consider the low rank matrix approximation

$$\boldsymbol{X}_r = \sum_{j=1}^{r} \boldsymbol{u}_j \sigma_j \boldsymbol{v}_j^T \tag{2}$$

with $r \leq R$. An important property of Xr is that it is the best rank r approximation to X in the least squares sense. In other words,

$$\boldsymbol{X}_r = \underset{\boldsymbol{Y}}{\text{argmin}} \|\boldsymbol{X} - \boldsymbol{Y}\| \quad \text{such that } \text{rank}(\boldsymbol{Y}) = r,$$

with respect to both the l_2 and Frobenius norms. Further, the relative error in this rank-*r* approximation using the l_2 norm is

$$\frac{\|\boldsymbol{X} - \boldsymbol{X}_r\|_{l_2}}{\|\boldsymbol{X}\|_{l_2}} = \frac{\sigma_{r+1}}{\sigma_1} \tag{3}$$

From (3), we immediately see that if the singular values decay rapidly, $(\sigma_{j+1} \ll \sigma_j)$, then $\boldsymbol{X}_r$ is a good low-rank approximation to $\boldsymbol{X}$. This property makes the SVD a popular tool for compressing data.

2.2. Unsupervised learning: *K*-means clustering

The *k*-means algorithm is iterative, first assuming initial values for the mean of each cluster and then updating the means until the algorithm has converged. the algorithm proceeds as follows:

1. given initial values for k distinct means, compute the distance of each observation xj to each of the k means.

2. Label each observation as belonging to the nearest mean.

3. Once labeling is completed, find the center-of-mass (mean) for each group of label points. These new means are then used to start back at step (1) in the algorithm.

2.3. Unsupervised learning: Gaussian Mixture Models

The basic assumption in this method is that data observations xj are a mixture of a set of k processes that combine to form the measurement. Like k-means and hierarchical clustering, the GMM model we fit to the data requires that we specify the number of mixtures k and the individual statistical properties of each mixture that best fit the data. GMMs are especially useful since the assumption that each mixture model has a Gaussian distribution implies that it can be completely characterized by two parameters: the mean and the variance.

2.4. Supervised learning: Linear Discriminant Analysis

The LDA algorithm aims to solve an optimization problem to find a subspace whereby the different labeled data have clear separation between their distribution of points. This then makes classification easier because an optimal feature space has been selected. The goal of LDA is two-fold: find a suitable projection that maximizes the distance between the inter-class data while minimizing the intra-class data. A general projection could lead to very poor discrimination between the data. LDA can separates the probability distribution functions in an optimal way.

2.5. Supervised learning: Support Vector Machines

Linear SVM: The optimization problem associated with SVM is to not only optimize a decision line which makes the fewest labeling errors for the data, but also optimizes the largest margin between the data.

Nonlinear SVM: In order to build model of restrictive and high-dimensional space data, the feature space for SVM must be enriched. Nonlinear SVM maps the data into a non-linear higher-dimensional space, so as to include nonlinear features and build hyperplanes in the new space. And the optimization of clusters is made in the higher-dimensional new space.

Kernel Methods for SVM: The higher-dimensions in nonlinear SVM leads to an expensive computation. Kernel function allows us to represent Taylor series expansions of a large (infinite) number of observable in a compact way. The kernel function enables one to operate in a high-dimensional, implicit feature space without ever computing the coordinates of the data in that space, but rather by simply computing the inner products between all pairs of data in the feature space.

2.6. Neural networks and Deep Learning

The NN can be multi-layers, the activation functions in each layer should be chosen. Importantly, the chosen activation function will be differentiated in order to be used in gradient descent algorithms for optimization. Many functions like, linear, binary step, logistic, TanH, rectified linear unit (ReLU) are either differentiable or piecewise differentiable.

The NN is built when all weights of the network are determined. In practice, the objective function chosen for optimization is a proxy due to the ability to differentiate the objective function. A loss function can be chosen so as to approximate the true objective. The backpropagation (Backprop) algorithm exploits the compositional nature of NNs in order to frame an optimization problem for determining the weights of the network. Backprop relies on a simple mathematical principle: the chain rule for differentiation.

Backprop allows for an efficient computation of the objective function's gradient. While, another algorithm is also critical for enabling the training of NNs, which is stochastic gradient descent (SGD). SGD provides a more rapid evaluation of the optimal network weights. Since SGD does not estimate the gradient using all n data points. Only a single, randomly chosen data point, or a subset for batch gradient desent, is used to approximate the gradient at each step of the iteration.

Deep convolutional neural networks (DCNN) is fundamental building block of deep learning methods. The prototypical structure of a DCNN includes convolutional layers(each convolution window transforms the data into a new node through a given activeation function) and pooling layers (to progressively reduce the spatial size of the representation in order to reduce the number of parameters and computation in the network).

NN can also be used to predict the future state of dynamical systems. The goal is to train a NN to learn an update rule which advances the state space from x_k to x_{k+1}, where k denotes the state of the system at time t_k. The NN is trained with input and output data, while, input data is a matrix of the system at x_k and the output is the corresponding state of the system x_{k+1} advanced Δt.

2.7. Dynamic Mode Decomposition (DMD)

DMD assumes that data is generated by an unknown dynamical system so that the columns of $\boldsymbol{X}$, $\boldsymbol{x}(t_k)$, are time snapshots related by the map $\boldsymbol{x}(t_{k+1}) = \boldsymbol{F}(\boldsymbol{x}(t_k))$. While $\boldsymbol{F}$ may be nonlinear, the goal of DMD is to determine the best-fit linear operator $\boldsymbol{A}: \mathbb{R}^m \rightarrow \mathbb{R}^m$ such that

$$\boldsymbol{x}(t_{k+1}) \approx \boldsymbol{A}\boldsymbol{x}(t_k)$$

If the two time-shifted data matrices are defined as follow,

$$\boldsymbol{X}_1^{n-1} = \left[\begin{array}{cccc} | & | & \cdots & | \\ \boldsymbol{x}(t_1) & \boldsymbol{x}_2(t_2) & \cdots & x(t_{n-1}) \\ | & | & \cdots & | \end{array}\right],$$

$$\text{and } \boldsymbol{X}_2^{n} = \left[\begin{array}{cccc} | & | & \cdots & | \\ x(t_2) & x(t_3) & \cdots & x(t_n) \\ | & | & \cdots & | \end{array}\right],$$

then we can equivalently define $\boldsymbol{A} \in \mathbb{R}^{m\times m}$ to be the operator such that

$$\boldsymbol{X}_2^n \approx \boldsymbol{A}\boldsymbol{X}_1^{n-1}.$$

It follows that A is the solution to the minimization problem

$$\boldsymbol{A} = \min_{\boldsymbol{A}'} \left\|\boldsymbol{X}_2^n - \boldsymbol{A}'\boldsymbol{X}_1^{n-1}\right\|_F,$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

A unique solution to this problem can be obtained using the *exact DMD* method and the Moore-Penrose pseudo-inverse $\hat{\boldsymbol{A}} = \boldsymbol{X}_2^n \left(\boldsymbol{X}_1^{n-1}\right)^{\dagger}$. Alternative algorithms have been shown to perform better for noisy measurement data, including optimized DMD [3], forward-backward DMD, and total-least squares DMD.

2.8. Koopman Operator Theory

Koopman theory is that any nonlinear dynamical system could be represented in terms of an infinitedimensional linear operator. In practical, obtaining finite-dimensional, matrix approximations of the Koopman operator is the focus of intense research efforts and holds the promise of enabling globally linear representations of nonlinear dynamical systems.

The Koopman operator is linear, which is appealing, and a prmary motivation to adopt the Koopman framework is the ability to simplify the dynamics through the eigendecomposition of the operator.

However, obtaining Koopman eigenfunctions from data or from analytic expressions is a central applied challenge in modern dynamical systems. A set of Koopman eigenfunctions may be used to generate more eigenfunctions like the first question in this homework.

3. Algorithm Implementation and Development

3.1. Standard DMD

Arrange the data $\{z_0, \ldots, z_m\}$ into matrices.

$$X \triangleq \left[\begin{array}{ccc} z_0 & \cdots & z_{m-1} \end{array}\right], \quad Y \triangleq \left[\begin{array}{ccc} z_1 & \cdots & z_m \end{array}\right]. \quad (4)$$

Compute the (reduced) SVD of X, writing.

$$X = U\Sigma V^* \quad (5)$$

where U is $n \times r$, Σ is diagonal and $r \times r$, V is $m \times r$, and r is the rank of X. Define the matrix

$$\tilde{A} \triangleq U^* Y V \Sigma^{-1} \quad (6)$$

Compute eigenvalues and eigenvectors of $\tilde{A}$, writing

$$\tilde{A}w = \lambda w \quad (7)$$

The DMD mode corresponding to the DMD eigenvalue λ is then given by

$$\hat{\varphi} \triangleq Uw \quad (8)$$

3.2. Least square curve fitting

In the 3rd question of part 1, we need to use data to fit values of 4 parameters in Lotka-Volterra equations. Here are some steps of this algorithm.

1. Define the Lotka-Volterra equations in a form ready for solution by ode45. The function is *LVequation()*.

2. Define the variable r for the solution of 4 parameters. In this question, r is 4×1 matrix.

3. Since the objective function for this problem is the sum of squares of the differences between the ODE solution with parameter r and the provided population data. In order to express the objective function, in this step, we define the function that computes the ODE solution using parameters r. This function is *solpts = RtoODE(r,tspan,y0)*.

4. To use RtoODE in an objective function, convert the function to an optimization expression by using fcn2optimexpr.

5. Express the objective function as the sum of squared differences between the ODE solution and the solution with true parameters.

6. Create an optimization problem with the objective function obj. And view the problem.

7. Give the initial guess *r*0 for the solver and call solve.

8. Use the solution to reconstruct the data and compare with original data.

3.3. SINDy with interpolation

Since the population data is given by years, the results of SINDy is not correct due to the derivatives are computed with nearby points. In order to avoid it, the interpolation method is used before SINDy. Spline interpolation using not-a-knot end conditions. The interpolated value at a query point is based on a cubic interpolation of the values at neighboring grid points in each respective dimension.

4. Computational Results

4.1. Part1: Models of population data

There are five methods used to find the model of the population of lynx and snowshoe. The time-delay DMD model is the best fit nonlinear, dynamic systems model to the given data. Next, the results of the models are presented.

4.1.1. OptDMD model and Bagging.

DMD is based on proper orthogonal decomposition (POD), which utilizes the computationally efficient singular value decomposition, so that it scales well to provide effective dimensionality reduction in highdimensional systems. DMD not only provides dimensionality reduction in terms of a reduced set of modes, but also provides a model for how these modes evolve in time. The first step is to reshape the snapshots of the state of a system evolving in time to two matrices. $\mathbf{X}$ and $\mathbf{X}'$. The DMD algorithm seeks the leading decomposition (eigenvalues and eigenvectors) of the best-fit

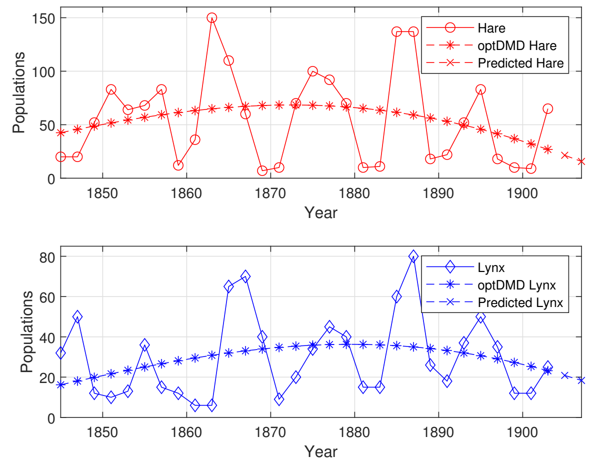


Figure 1. The optDMD model to forecast the future polulation states.

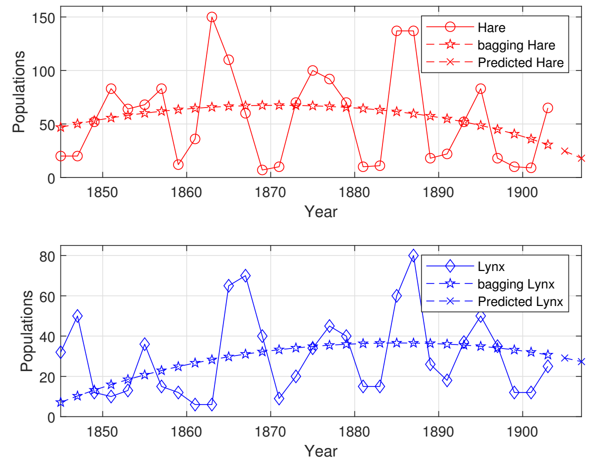


Figure 2. Bagging optDMD model to forecast the future polulation states.

linear operator A that relates the two snapshot matrices in time: $\mathbf{X}' \approx A\mathbf{X}$. The four steps and the method to write the spectral expansion in continuous time can be found in chapter 7.2 in book [1].

The standard DMD algorithm has been described above, because the teacher gave the optDMD code, so directly apply optDMD to build the prediction model. The results are shown in Figure 1. The upper graph is the result graph of the optDMD model, and the lower graph is the result graph after bagging the optDMD model. Predicted population for the next 4 years is represented in the figure with different legends. The prediction results of optimal DMD is not satisfying, since the decay due to the real part of Ω is too strong, the predicted model failed after the known input first data.

The bagging algorithm is a technique for obtaining S new data sets after selecting S times from the original data set, which is a sampling with replacement. Suppose there

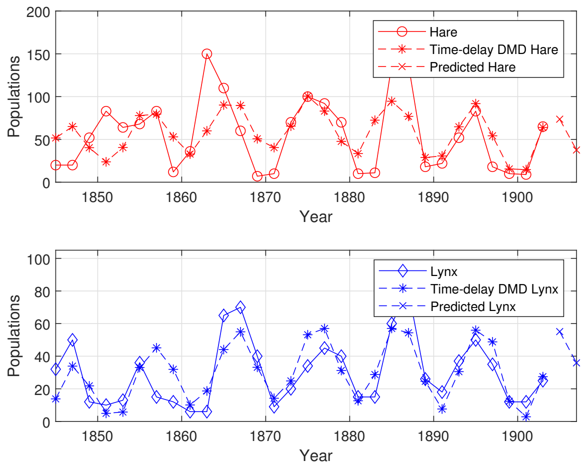


Figure 3. The time-delay optDMD model.

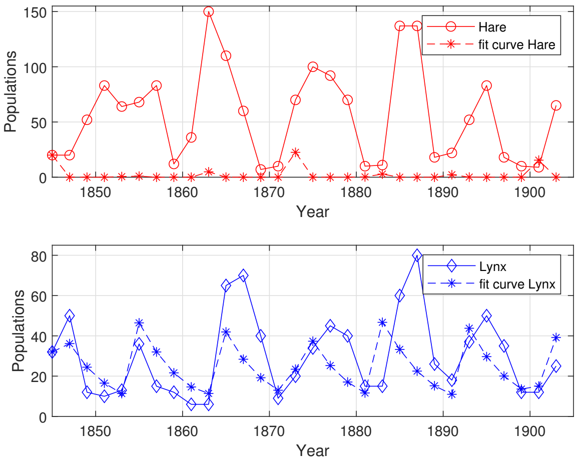


Figure 5. Curve fitting result.

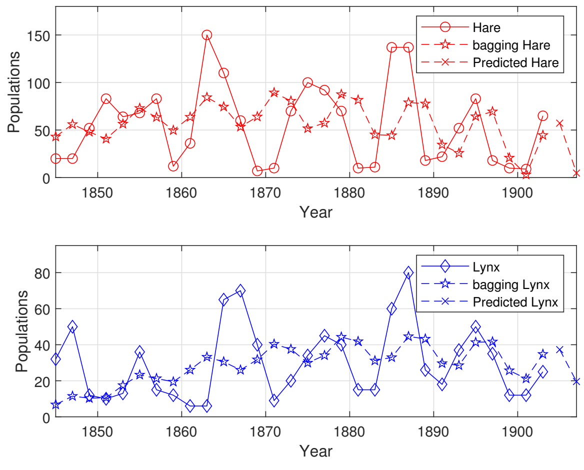


Figure 4. Bagging time-delay optDMD model.

is a data set D, using the random sampling method with replacement to take k data subsets (the number of subset samples are all equal): D1, D2, ..., Dk, as a new training set, use these k subsets The sets are trained separately using algorithms such as classification and regression, and finally k models are obtained. The collection strategy of bagging is also relatively simple. For classification problems, a simple voting method is usually used, and the category or one of the categories with the most votes is the final model output. For regression problems, the simple average method is usually used to arithmetically average the regression results obtained by k weak learners to obtain the final model output. I also tryied bagging the optDMD to improve the results as shown in figure 2. Unfortunately, the predictions are not very well.

4.1.2. Time-delay optDMD model and Bagging.

Time-delay DMD model is applied with 5 times delay coordinate, as shown in Figure 3. Other times delay coordinates are also applied, but the effect is very different, so 5 times is relatively good. In this time-delay coordinate, the Ξ shows that we can reconstruct the data with top 26 modes. Compared the DMD model with original data, it shows better prediction than optDMD model. And there are 3 elements of ω with zero real parts and other elements are symmetrically distributed. I also tryied bagging the time-delay optDMD model with 100 subsections, but the results turned out to be even worse as shown in figure 4.

4.1.3. Least Square Curve Fitting.

Since the Lotka-Volterra equations are provided, we can use the least square curve fitting method to find the best fit parameters of b, p, r and d. Solving problem using *lsqnonlin()*, and the fit values are:

b = 6.487838
p = 0.262464
r = 0.103026
d = 0.195660

In Figure 5, the prediction model is somewhat oscillating, with the graph showing the population change curve drawn by the fitted parameters.

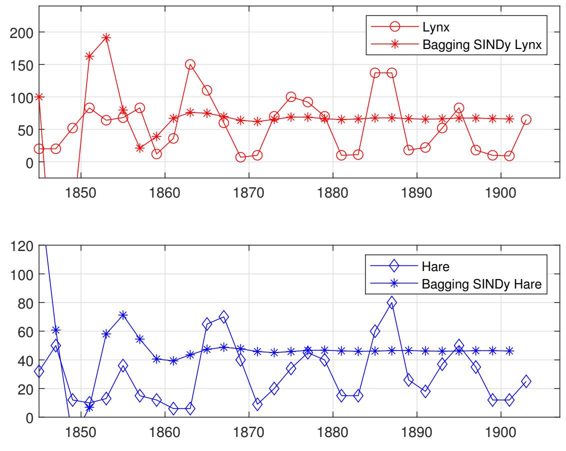
It can be concluded that the curve does not accurately fit the original data. And the results are sensitive to the initial values of the parameters set, and different initial values may lead to completely different fitting results. Here the initial values of b, p, r and d are [5 0.3 0.1 0.3]. In conclusion, this curve fitting method outperforms optDMD, but does not perform well compared to the time-delay DMD model. So, even knowing a similar equation that models a population phenomenon, one cannot expect to be able to build a model from the raw data with this equation.

4.1.4. Bagging sparse regression model SINDy.

The SINDy leverages the fact that many dynamical systems have dynamics f with only a few active terms in the space of possible rignt-hand side functions. We seek to approximate f by a generalized linear model with the

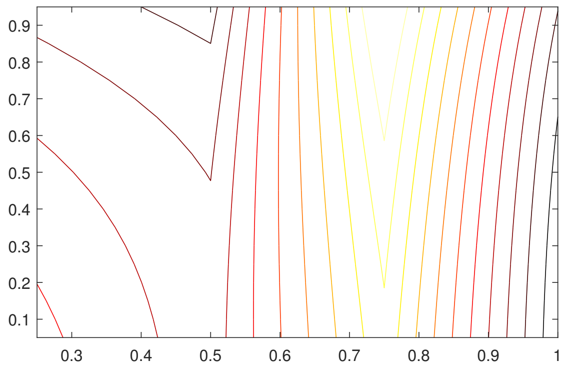
```
{0×0 char}    {'xdot'              }    {'ydot'              }
{'1'      }   {[19.073247082752911]}    {[-1.392277998536151]}
{'x'      }   {[                 0]}    {[ 0.150147354528584]}
{'y'      }   {[-0.604086380268193]}    {[-0.234933117558625]}
{'xx'     }   {[                 0]}    {[                 0]}
{'xy'     }   {[                 0]}    {[                 0]}
{'yy'     }   {[                 0]}    {[                 0]}
{'xxx'    }   {[                 0]}    {[                 0]}
{'xxy'    }   {[                 0]}    {[                 0]}
{'xyy'    }   {[                 0]}    {[                 0]}
{'yyy'    }   {[                 0]}    {[                 0]}
```

(a) Parameter value

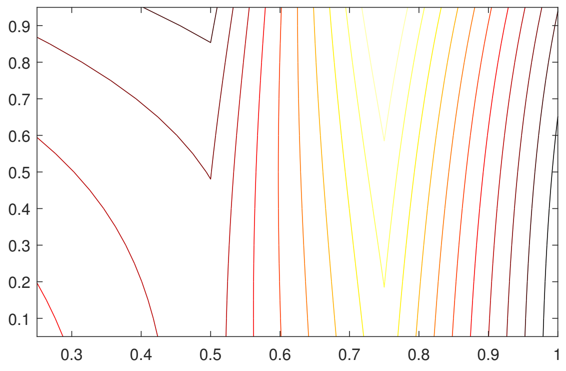


(b) Curve fitting result

Figure 6. Bagging sparse regression SINDy results.



(a) ODE result



(b) NN result

Figure 7. NN for KS equation when X has 4 states.

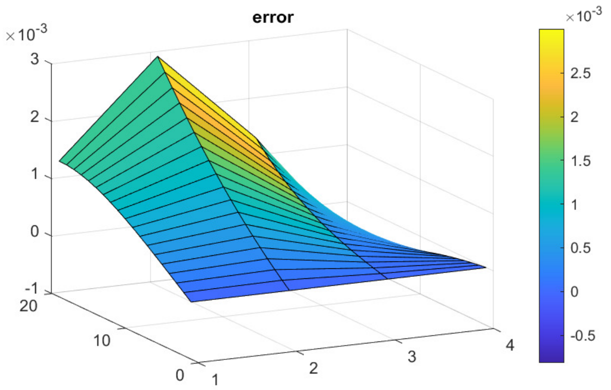


Figure 8. Error between NN and ODE.

fewest nonzero terms in ξ as possible. A library of candidate nonlinear function $\Xi(X)$ may be constructed from data X.

$$\frac{\mathrm{d}}{\mathrm{d}t}x = \mathbf{f}(\mathbf{x}) \tag{9}$$

$$\mathbf{f}(\mathbf{x}) \approx \sum_{k=1}^{p} \theta_k(x)\xi_k = \Theta(x)\xi \tag{10}$$

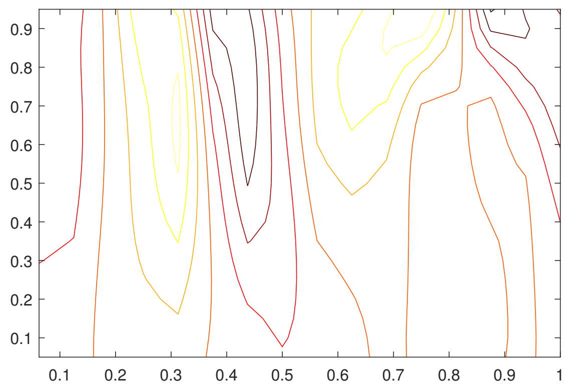
In order to find the bagging sparse regression model, SINDy method is applied. However, the results is way far from the original data due to the derivatives computed with large time interval data. In order to find the correct model, interpolation is used before SINDy. The curious thing is that in the bagging sparse regression model, it has constant term, but without the term *xy*, it is kind of strange. The bagging sparse regression model is better than the basic DMD, but after time 20, it keeps constant due to the constant term and without the term *xy* in sparse regression model.

4.2. Part2: NN for Kuramoto-Sivashinsky equation and reaction-diffusion system of equations

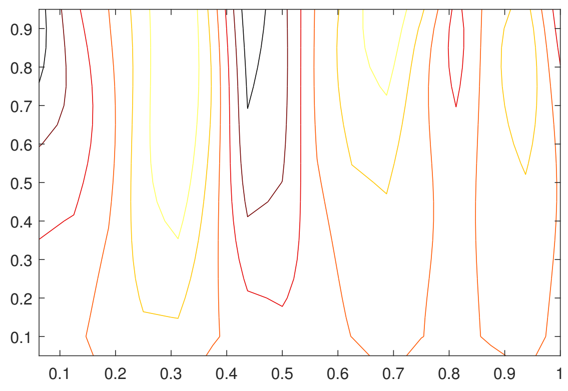
Neural Networks (NN) can be regarded as a map from input to output. The NN is built on trained data and validated with withdraw data, and used to predict with new data. Since the NNs have significant potential for overfitting of data, the cross-validation must be considered. NNs offer an amazingly flexible architecture for performing a diverse set of mathematical tasks, for instance, it can be used for future state predictions of dynamical systems.

4.2.1. Train a NN for the KS equation.

In this subsection, a NN is trained from usave(t) to



(a) ODE result



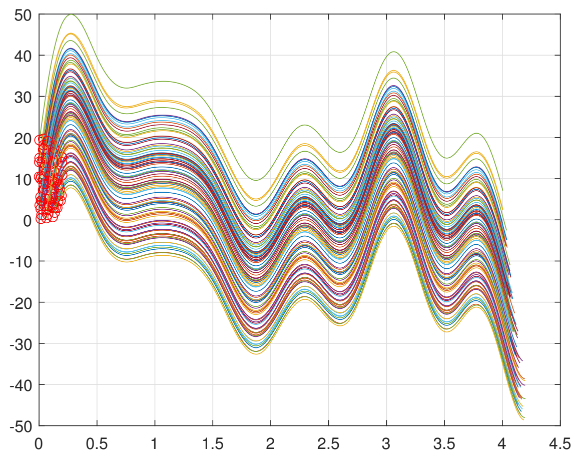
(b) NN result

Figure 9. NN for KS equation when X has 16 states.

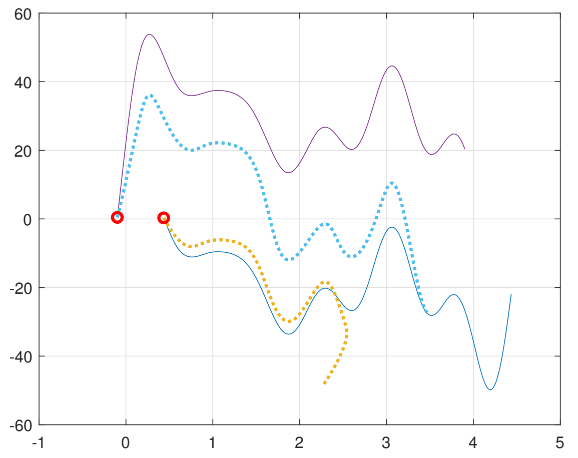
usave(t+1), and X is only in four states. The result of NN is shown in figure 7(b), and we can see that it is similar with the ODE result in figure 7(a). Figure 8 shows the error in space between them. But the issues is that the NN works not so well when the points are denser and X has more states, for example 16 states, as shown in figure 9. When I made the state of X more, the results were even more dissimilar. And due to the limitation of computer memory, it takes a lot of time to perform large-scale calculations. I don't know what is the essential parameters, but it is certain that the range of X and t of the NN training set will affect the effectiveness of the neural network model.

4.2.2. Compare of the evolution trajectories.

The evolution trajectories for my NN against using the ODE time-stepper is compared, the result is shown in figure 10. Because when I train the solution of the KS equation in NN, the better result is that X has 4 states, so it is not ideal when drawing the training trajectory. And the KS equation is an implicit equation, not similar to the Lorentz equation. So in the end the trajectory graph I show is the trajectory graph with the u expression fixed. u = -sin(x)+2cos(2x)+3cos(3x)-4sin(4x). It can be seen that the



(a) ODE trajectories



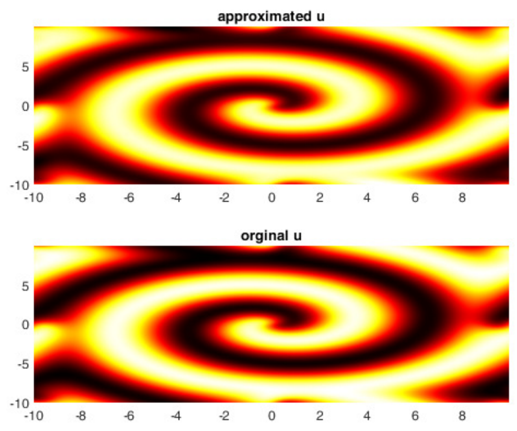
(b) NN trajectories

Figure 10. Compare trajectories of NN with ODE.

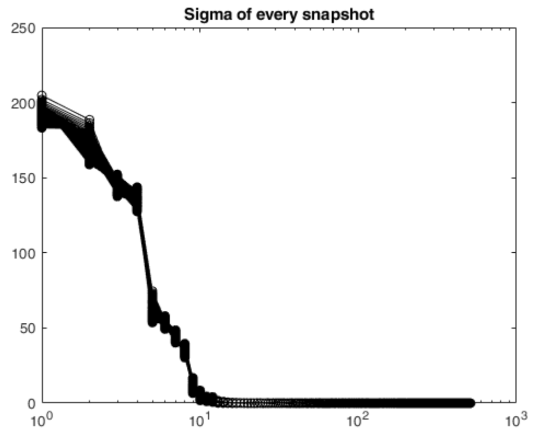
evolution trajectories of NN is similar. Using deep learning algorithms such as neural networks to find numerical solutions of ordinary differential equations allows the network to learn and update the optimization parameters, which greatly reduces the mathematical difficulty in the modeling process.

4.2.3. Low-dimensional subspace via the SVD for reaction-diffusion system.

In many signal processing applications, e.g. pattern recognition, linear regression, image enhancement, and so on, there are a huge amount of data with very high dimensions. Dealing with these high dimensional data makes the processing tasks very difficult and in some applications such as linear regression and variable selection, one can not have a clear interpretation of the signals at hand. However, despite of their very high dimensions, it is well known that natural signals such as images and speech signals actually

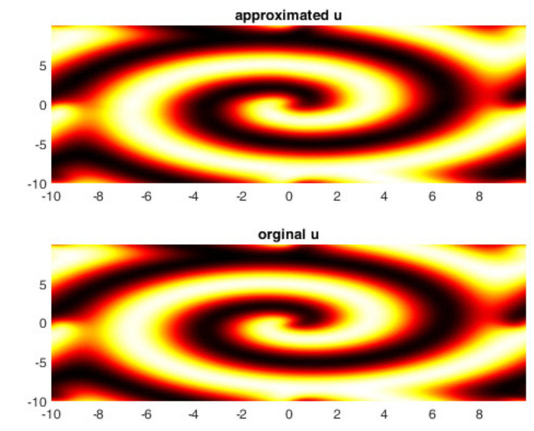


(a) Low-rank reconstruction and original u

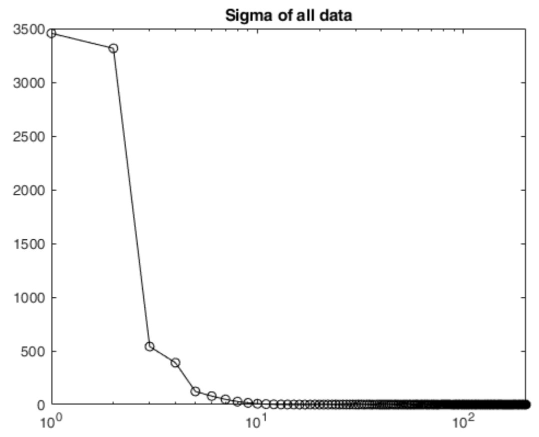


(b) Sigma of each snapshot

Figure 11. Low-rank model.



(a) Low-rank reconstruction and original u



(b) Sigma of All data

Figure 12. Low-rank model with all data.

lie in a low dimensional linear subspace. In other words, the intrinsic dimension of many natural signals is much less than their length.

Singular value decomposition (SVD) is one of the most powerful and ubiquitous algorithms for transforming and simplifying data, enabling data compression for efficient sensing and compact representation for modeling and control. Despite rapid improvements in the resolution of measurements and computations, many complex systems still exhibit dominant low-dimensional patterns in the data.

In this section, the reaction-diffusion system of equations is projected into a low-rank space via the SVD . In Figure 11, each snapshot was taken SVD, and the 10 modes are used to construct the original data u. In Figure 12, all data used to find the low-rank space, and the dominant rank is 4 when using all data. Here I have the same problem when training NN. It doesn't work when I have such big data scale, even I reduce it using SVD, and I reshape U, V and Ξ together as one matrix, and use it as input and output. But MATLAB didn't pop up the NNtools.

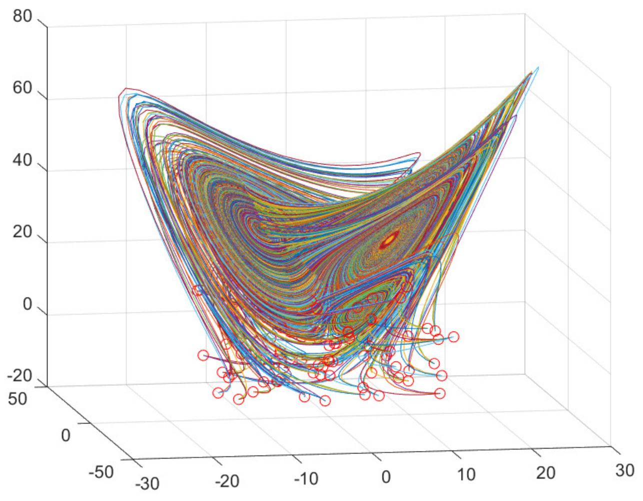


Figure 13. Training data.

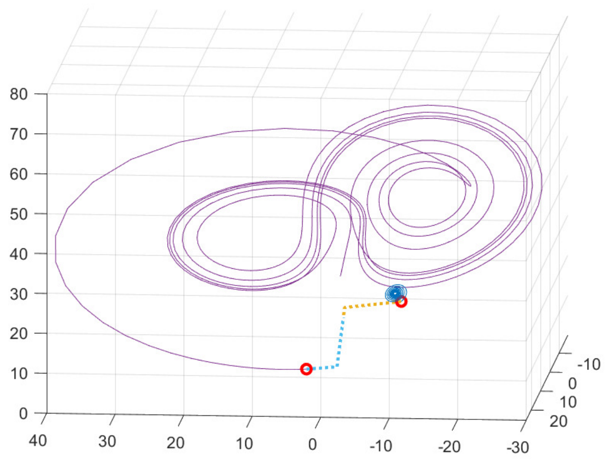


Figure 14. Trajectories of NN and ODE.

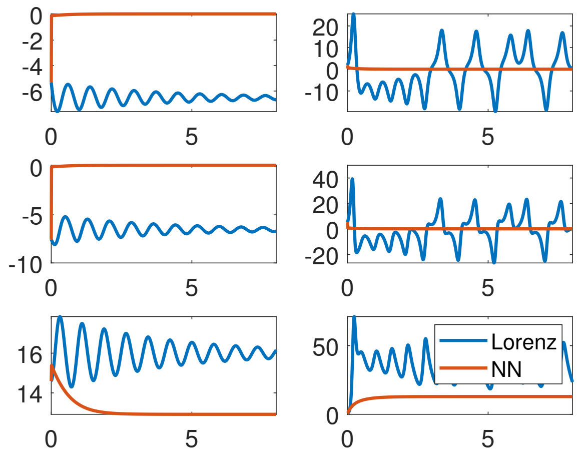


Figure 15. Comparison of NN and ODE.

4.3. Part3: NN training for three different Lorenz equation

One NN is trained with 3 sets of data from three different Lorenz equation, and each sets contains 100 trajectories. Of course, these three sets have ρ are 10, 28 and 35, respectively. The training data trajectories are shown in Figure 13. The neural network trajectory is shown in Figure 14. Their comparison is shown in Figure 15. It turns out that the NN can not predict the new data from equation with new ρ. If more number of ρ could cover the desired ρ 17 and 40, maybe the results will be better.

5. Summary and Conclusions

In this course, we learned many method to build a model from data of dynamic system. Some of them are quite useful when dealing with data without any governing equations. Some methods like SVD, DMD, SINDy, Koopman operator, Networks, Bagging, and time-delay coordinate are good tools to explore the data from measurement or simulation.

Besides, these tools for dynamic system. The architecture of NN and deep learning were explained very well. These method are also quite useful. We can also get many resources like the book, website and videos to learn more. Due to my limited time, I couldn't perfectly understand and apply all the knowledge, but I was inspired a lot both in class and during my homework. Hope that I could use some of ideas for my research in the future.

References

[1] S. L. Brunton and J. N. Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

[2] T. Askham and J. N. Kutz. *Variable projection methods for an optimized dynamic mode decomposition*. SIAM Journal on Applied Dynamical Systems, vol. 17, no. 1, pp. 380-416, 2018.