Tuesday, November 6, 2018

HW5: Project Functional & Component Specifications

Functional Specifications

- Background
  - Dynamical systems are important concepts, but the way to find the corresponding models are elusive. Also, even if we find them, nonlinear dynamical systems are hard to analyze. Thanks to the advance of the computational power and the abundance of data nowadays, we now have the opportunities to uncover the governing law of the dynamics in a data-driven manner and apply global linearization based on Koopman theory. Here, we provide the implementations of some pioneer works in this community. (In fact, you can find these pieces of codes elsewhere, but in different places with different programming languages and styles. We provide them in a unified framework, with proper tests and organizations, all in one place.)
- User Profile
  - We aim at serving three types of users:
    - *Researchers / experimentalists who work with dynamics data in a scientific field, barely know statistics or computer science but have domain expertises.*
    - *Data scientists who are equipped with statistical weapons but lacking insights in dynamical systems.*
    - *Researchers who wants to enter this field. This is data-driven dynamical system 101. (Basic applied math, algebra, statistics, computer science backgrounds required.)*
- Data Sources
  - *Toy data for demo purposes (simulate some differential equations)*
  - *Large-scale fluid dynamics simulations (Navier-Stokes)*
  - *Global climate data*
  - *Belousov-Zhabotinsky reaction experimental data*
- Use Cases
  - USE CASE 1:

    - *objective: Identify differential equations from data*
    - *interaction between user & system: user inputs the time series data and some other optional parameters, the system outputs a differential equation*
  - USE CASE 2:
    - *objective: Linearize nonlinear dynamics*
    - *interaction between user & system: user inputs the simulation data from a nonlinear dynamical system, the system outputs the coordinate system that linearize the dynamics, together with the linear dynamics itself.*
  - USE CASE 3:
    - *objective: Visualize bifurcation in the phase space (for 2D systems)*
    - *interaction between user & system: user inputs two time series (two observables of a system evolve in time), the system output an interactive phase space plot, which the user can adjust the system parameters interactively.*

Component Specifications

- Software Components
  - Data Manager
    - *Function: Read or generate the data and apply necessary manipulations inside the data manager, eg. smoothing, differentiation, time-delay embedding*
    - *Input: a url, a file name (read from file) or a system name (eg. Lorenz)*
    - *Output: a preprocessed dataframe*
  - Model Manager
    - *Function: pick a suitable model with specified parameters (Models might include: SINDy, ISINDy, HAVOK, DMD, EDMD, Kernel DMD, MrDMD)*
    - *Input: the model to use and the hyper-parameters*
    - *Output: either a fitted equation or the linearized system (with the coordinate basis)*
  - Visualization Manager
    - *Function: Compare reconstructions & predictions; visualize bifurcation diagram*

    - *Input: specify the method and the dataset(s)*
    - *Output: a plot (either interactive or not)*
- Interactions to Accomplish use cases
  - USE CASE 1:
    - *User inputs data to the Data Manager and gets output (governing equation) from Model Manager (or the Visualization Manager if he wants the plots!)*
  - USE CASE 2:
    - *User inputs data to the Data Manager and gets output (the linearize system and the coordinates) from Model Manager*
  - USE CASE 3:
    - *User inputs data to the Visualization Manager and gets output (an interactive plot) also from the Visualization Manager*
- Preliminary Plan (In priority order)
  - Build up a pipeline with the most basic datasets, preprocessing steps, algorithms and visualizers. Ignore all other details.
    - *dataset: small scale simulations (eg. Harmonic Oscillator, Van der Pol Oscillator)*
    - *preprocessing step: time-delay embeddings*
    - *algorithms: SINDy (Sparse Identification of Nonlinear Dynamics), DMD (Dynamic Mode Decomposition)*
    - *visualizer: line plots, bifurcation diagram with 'widget'.*
  - Analyze the second real-world dataset (not simulations!): global climate data.
  - Design Unit tests for each software components.
  - Write 1~2 Ipython Notebook Demo for tutorial purposes.
  - Adding other advance algorithms such as ISINDy (implicit-SINDy), HAVOK(Hankel Alternative View of Koopman), EDMD (Extended DMD), etc.
  - Design another round of Unit Tests.