Project: A new methodology for comparing performance of prediction algorithms for chaotic dynamical systems

Instructors: Krishna Palem and Robert Cartwright
Project supporters: Devika Subramanian and Nikola Jovanovic
Technical participant: Adam Duracz

1 The problem and current state of the our work

- Our project involves developing a statistically sound methodology for comparing the performance of three machine learning algorithms (ESN (echo-state networks), LSTM (long-short term memory) and ensembles of ESNs) on predicting trajectories of four interesting chaotic dynamical systems: Lorenz96, Lorenz63, Lorenz40 and the Kuramoto-Sivashinsky (KS) system.
- The first goal is to establish which of the methods perform best on each of these benchmarks. The second goal is to understand why each method performs as it does on these benchmarks.
- We have completed the analysis for Lorenz96. Our results are in the folder `lorenz_96, 200, 1`. All steps in the analysis pipeline are outlined in the next section.
- Your task is to run Lorenz63, Lorenz40 and the KS system through the same analysis pipeline and produce results on the performance of the three learning algorithms.

2 The Analysis Pipeline

- Get the dataset for your system. All data resides in the folder named `data_sets`. The subfolders are `lorenz_96`, `lorenz_63`, `lorenz_40` and `ks` – each contains a `.csv` file with time series data on each system. For example, `lorenz_96.csv` is a file with a million lines where each line is a comma separated list of eight real values, corresponding to the value of the variables $X_1, \ldots, X_8$ of the Lorenz96 system evolving from time step 0 to time step 999999. This groud truth data will be used to generate training, validation and test sets for the models.
- To train an ESN model on default parameters with Lorenz63 and predict with it

```
from script.model import Model, DataSet
from script.model_esn import ModelEsn
import math
import numpy as np
import os

data_set = DataSet('lorenz_63',0.906,1)
model_params = Model.default_model_params('esn')
run_params = Model.default_run_params()
model = ModelEsn(model_params,data_set.path_prefix())
model.load_or_train(data_set.data,True)
warm_up_input = data_set.data[:,model_params.tr_len:model_params.tr_len
              +run_params.warm_up_size]
pos_inside_run = model_params.tr_len+run_params.warm_up_size
preds = model.predict(warm_up_input,run_params,pos_inside_run,True)
```

- If you want to run another dataset with ESN, go into the `parameters` folder and change `default_esn.json` to adjust model parameters and `default_run.json` to adjust prediction or run parameters.
- You will need to find good parameter settings for ESN, LSTM, LSTM2, ESN_ensemble models. We have provided a basic parameter search functionality for ESNs in `esn_parameter_sweep.py` to find the model parameters that work best for a new dataset. Make sure you log the results of the parameter sweep. Once you know the best parameters, you can update the default model parameter file in `parameters`. You will need to write corresponding parameter sweep function for LSTMs and ESN_ensemble.
- Compute prediction horizon statistics on a consecutive set of initial conditions. The code to do this is in `pred_horizon_stats.py`. You will need to uncomment the appropriate lines in function `main()` to correspond to the model type and ensure the right model and run parameters are set in the `parameters` folder. This step is needed to design a fair distribution of initial conditions to test each of the models. The function creates predictions from the chosen initial conditions and writes them into the folder defined by dataset name, MTU (Model Time Unit, time interval equivalent to 200 time-steps for Lorenz96 system), and training skip parameters.
- Perform model comparison by analyzing predictions from a pair of models using `combine_graphs.py`. All results (including jpgs of figures) are written into the folder defined by dataset name, MTU, and training skip parameters. You can check your results against the ones we generated for Loren96 with 200 MTU and skip of 1.
- To test robustness, compare the following model configurations
  - ESN with best discovered parameters
  - ESN with best discovered parameters trained on a different training set (default is the first 40000 samples)
  - ESN with best discovered parameters with different randomization parameters (seeds)
  - LSTM with best discovered parameters
  - LSTM with best discovered parameters trained on a different training set
  - Boilerplate ensemble model, averaging predictions of ESN-s mentioned above
- Examine all the model variations and their predictive performance, and identify the best model and run parameters for each dataset.

3 Steps for getting started

- Obtain the tar gzipped file `ESN-Methodology-Research.tar.gz` from me (devika@rice.edu).
- Familiarize yourself with the code base at a high level and make sure you understand the interfaces to functions by reading README.txt provided with the code base.
- Reconstruct our Lorenz96 results with the existing defaults by following the analysis pipeline described above with default settings for all parameters.
- Try systematically changing model learning parameters or evaluation protocol on Lorenz96 to build comfort with running the analysis protocol on a new dataset.

4 Project deliverables

For each of Lorenz63, Lorenz40 and KS,

- Finding the best model parameters for the dataset
  - Sweep the parameter space to find the best model parameters for ESN on the data set.
  - Sweep the parameter space to find the best model parameters for LSTM1 on the data set.
  - Sweep the parameter space to find the best model parameters for ESN_ensemble on the data set.
- Compute prediction horizon statistics for each of the four models using `pred_horizon_stats.py`
- Perform model comparison pairwise for the base models with the best parameters producing plots generated by `combine_graphs.py`
- Try the training variations (retaining model parameters but training on a different sub-sequence) and perform model comparison with these variants.

5 Reference literature

- Essential references
  - Ashesh Chattopadhyay, Pedram Hassanzadeh, Devika Subramanian:
    Data-driven prediction of a multi-scale Lorenz 96 chaotic system using deep learning methods: Reservoir computing, ANN, and RNN-LSTM (2019)
- Additional references
  - Jaideep Pathak, Zhixin Lu, Brian R. Hunt, Michelle Girvan, Edward Ott:
    Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data (2017)
  - Pantelis R. Vlachas, Jaideep Pathak, Brian R. Hunt, Themistoklis P. Sapsis, Michelle Girvan, Edward Ott, Petros Koumoutsakos:
    Backpropagation Algorithms and Reservoir Computing in Recurrent Neural Networks for the Forecasting of Complex Spatiotemporal Dynamics (2020)
  - Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott:
    Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach (2018)