Dynamics and Model Discovery

Diego Alba, AMATH 563, 5/2020

Abstract:

The goal of this assignment is to evaluate differnt data-driven model discovery techniques, as well as to implement statistics to compare and rank modelpredictions. Two different dataset will be used, time series of population dynamics and snapshots of chemical reactions.

Background

The first dataset, concerning Canadian lynx and snowshoe hare populations from 1845 to 1903, contains 30 pairs of datapoints. We will see that in order to use some of the methods we will have to interpolate between the data to generate more points. The second dataset contains 1200 images of the Belousov-Zhabotinsky chemical oscillator.

The main model discovery techniqiue in this assignment will be Dynamic Mode Decomposition (DMD). The idea here is to find the best linear approximation to the dynamical system by projecting into a low rank space that captures the most important feattues (dynamic modes) of the system. This allows for good near-future predictions, and beacuse of cheap computations, it can be performed real time.

In order to apply DMD to cases with a small amout of data, or where not all variables are know, one can do time delay embeddings. That is, to make new variables from the ones available by shifting them in time. Similarly, another embedding technique is to apply a library of functions, such as polynomials, to the original data. Both techniques are able to find more accurate dynamical modes. Moreover, these versions of DMD can be coupled with regularization schemas such as L1 in order to promote sparsity and make the dynamic modes more interpretable.

To compare results we use several statistics. Kullback–Leibler divergence (KL divergence) is related to the Shanon entropy as measure of information or suprise and compares the similarity between a probablity distribution and a reference probability distribution. Akaike information criterion (AIC) is a measure that tries to compare models by not only contrasting the errors but also the number of parameters. Lastly, Bayesian information criterion (BIC) is similar to AIC, but with a different penalty for the number of parameters That is, with AIC the penalty is 2k, whereas with BIC the penalty is ln(n)k.

```
HTML Settings, don't know how to remove these two blocks
```

```
Out[2]:
```

```
Out[3]:
```

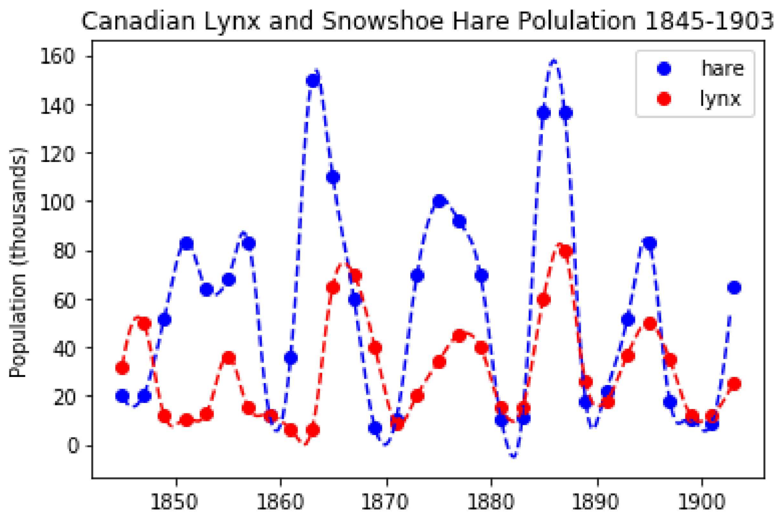
Click here to toggle on/off the raw code.

Implementation and Results

The fist thing we'll set up some settings for the Jupyter notebook and import all of the packages required to carry out the analysis and plotting.

We load the data and interpolate. This is depicted in Figure 1.

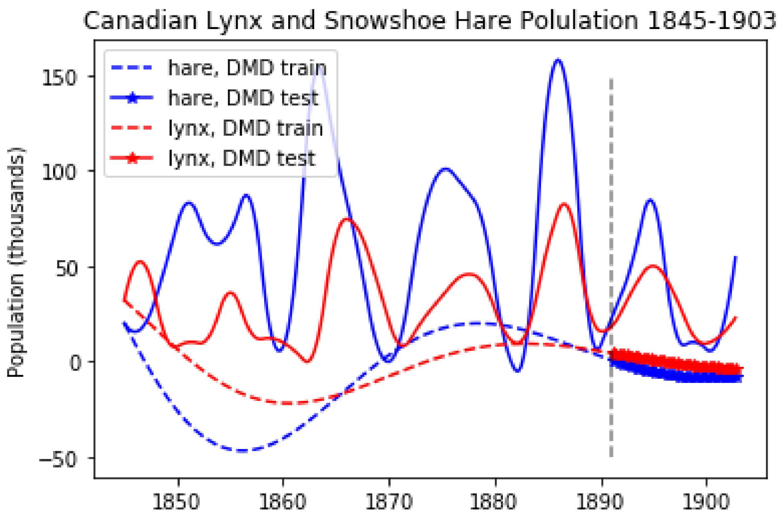
Figure 1



We go ahead and split the data into a train and test set. Because this is a time series and we would be interested in making predictions at future time points, we select the last 20% to be the test set (the split is denoted with a vertical line in the next figures).

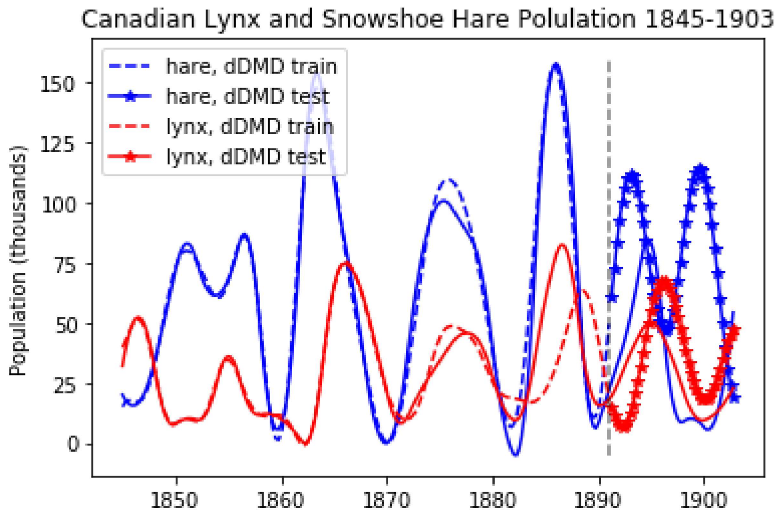
We use classis DMD to try to find the dynamic modes of the system. We are very limited as there's only data for two varibles and that is the dimensionality of our low rank space. In Figure 2, we can see how the model does oscillate but not with the right frequency or amplitude (it even goes negative). The predictions are also very poor.

Figure 2



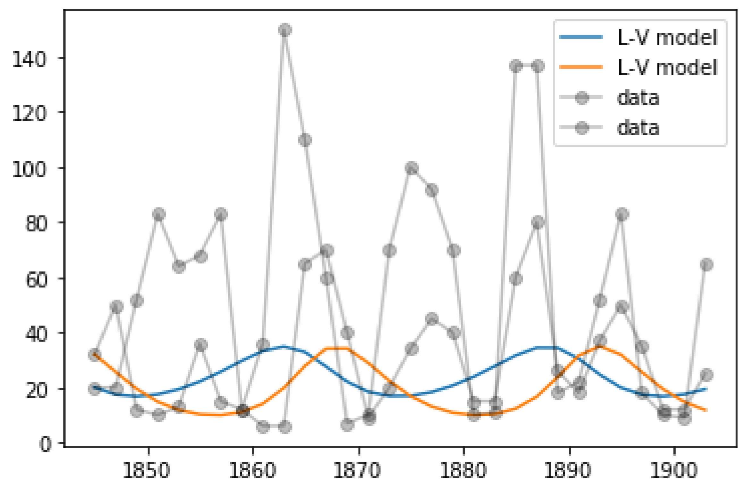
We try to imrpove these results by time-delay embedding. We make a total of 82 embeddings and do DMD on this data. In Figure 3 we can see the results are much better, both in terms of fitting to the training set and predicting the test set. We can see one the effects of using a low rank and linear approximation as the prediction gets worse over time.

Figure 3



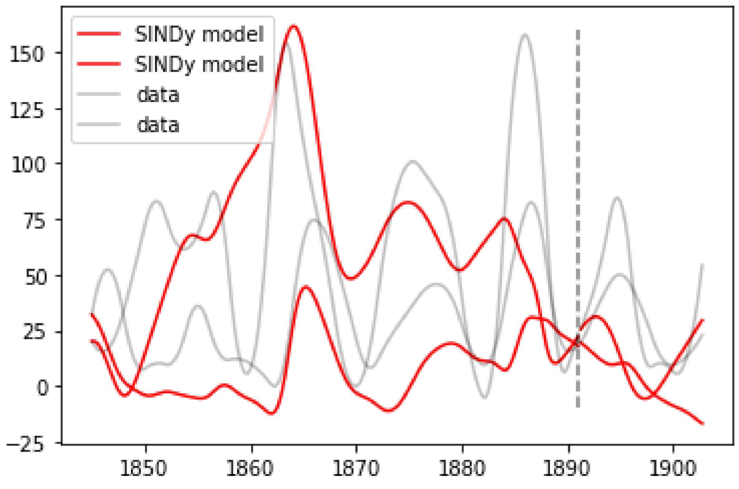
We also try to fit the data to the well-known Lotka-Volterra model, which uses 4 parameters that describe the interactions between species. We do this by defining the system of ODEs, integrating with scipy.odeint, calculating the square error between the data and model, and using scipy.minimize to drive the error down by changing the parameters. The best minimazation algorithm used was Nelder-Mead, but we can see in Figure 4 that this solution is worse that time-delayed DMD.

Figure 4



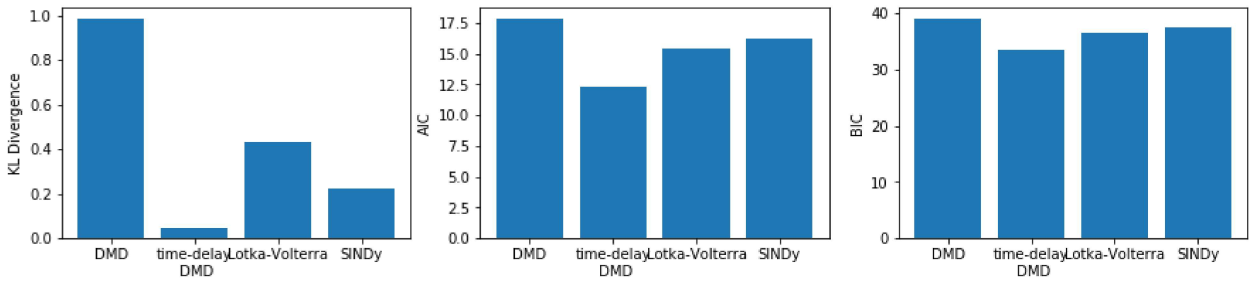
Lastly, we try to use sparse regression to fit a nonlinear, dynamical systems model to the data (SINDy). We do this by creating a library of functions (x,y,x*y,x*^2,*y*^2, *(xy)^2) and applying them to the population data. We can see this gives better results than fitting the Lotka-Volterra model, but worse than time-delay DMD.

Figure 5



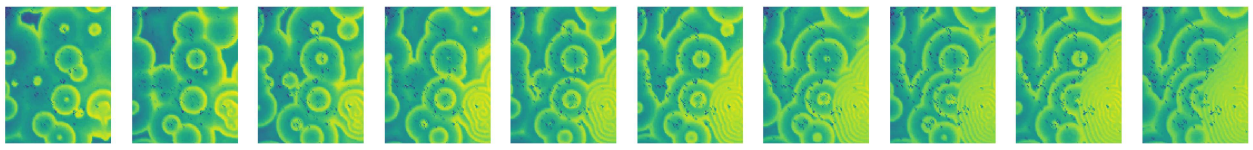
To compare across models, we calculate the population distribution accross all years for the data and each of the models, and calculate the KL divergence, AIC, and BIC scores. In Figure 6 we can clearly see that time-delay DMD gives the best results.

Figure 6



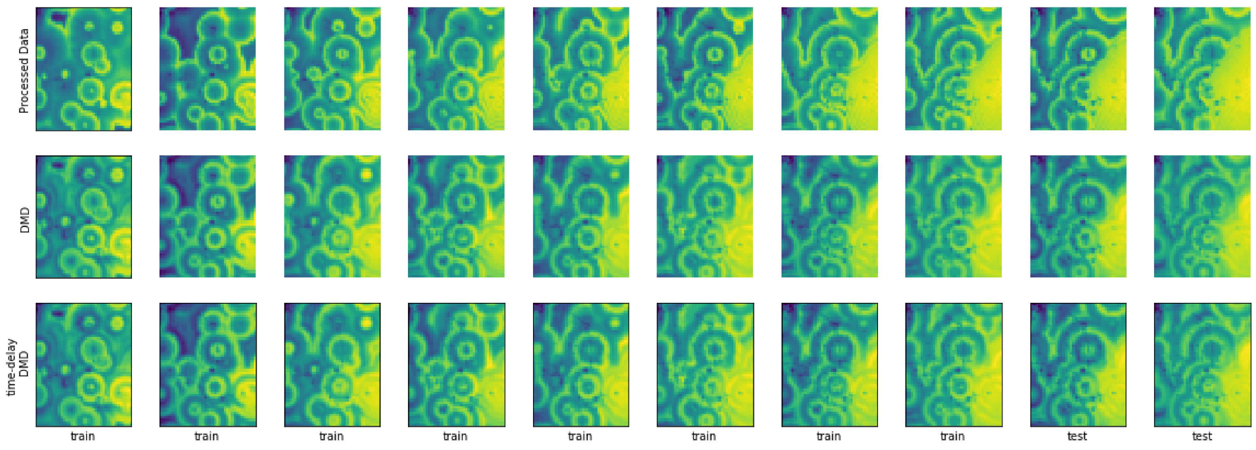
Switching gears to the Belousov-Zhabotinsky chemical oscillator dataset. We load the data and display 10 images at different time points in Figure 7.

Figure 7



Next, because the size of the images is too big, we reduce them to 50x39 and split it again into 80% train test and 20% test set. Moreover, we vectorize the images so at the end every row is a pixel and evry colum its evolution in time. We do DMD on the vecotrized images, and we also try time-delay embedding of 10. All the results and comparisson with the risized images is presented in Figure 8.

Figure 8



We can observe that in this case both DMD and time-delay DMD are very good, both when fitting to the training set and predicting the test set. This is porbably because the amount of data is much larger than in the Canadian lynx and snowshoe hare populations dynamics dataset.

Conlcusion

We have shown how to use different DMD variations for data-driven system identification for two very different datasets. Particularly, we have seen that the algorithm time-delay embedding is incredibly very useful when theres limited amounts of data or only a few variables have been measured. Moreover, we have shown how to compare moedls with different statistical score, such as KL divergence.

Appendix

The full Jupyter Notebook containing the report and the code used can be found at https://github.com/DIEGOA363/ISCS (https://github.com/DIEGOA363/ISCS)