Evaluating Current Machine Learning Techniques On Predicting Chaotic Systems

Sirapop Klinkachorn (sirapopk)
Stanford University Department of Computer Science

Jupinder Parmar (jsparmar)
Stanford University Department of Mathematics

Abstract

A chaotic system is one that shows large deviations in its behavior with small changes to its initial conditions. In particular we see this arise in the case of the kinematic motion of a double pendulum which we seek to model in order to understand more about predicting chaotic situations. We find that linear regression with a polynomial feature map is able to accurately model a double pendulum at small initial angles where chaotic motion is not yet apparent. For large angles, where the motion becomes chaotic, we discover that a Long Short-Term Memory Network can accurately predict the future trajectory of the system.

Introduction and Background

Chaotic systems are particularly interesting as they are especially sensitive to initial conditions and don't exhibit any recurring patterns within their behavior. Current research into systems governed by chaos theory has demonstrated that we still don't fully understand how these situations arise even if we are able to determine laws that describe them. Thus, finding a way to model these systems accurately is of crucial importance and would give us further insight into how these situations are formed. If we can apply machine learning to understand these relationships then there would be tremendous potential to not only figure out chaotic systems but understand the capabilities of machine learning models in general. There are a variety of physical phenomena that exhibit chaotic motion such as a duffing oscillator and a double pendulum. Within this paper we will specifically focus on understanding the chaotic kinematic motion of a double pendulum.

The Double Pendulum system is a widely used system to study non-linear behaviour and chaotic phenomena. For smaller angles, the behaviour of the pendulum can be well approximated with a polynomial function. However, for larger angles, there are no closed solutions that can describe the trajectory of the pendulum. The angle of the pendulum with respect to time can be derived by the first principle as follows.

$$\dot{\theta}_1 = \frac{6}{ml^2}\frac{2p_{\theta_1} - 3\cos(\theta_1-\theta_2)p_{\theta_2}}{16-9\cos^2(\theta_1-\theta_2)}$$

$$\dot{\theta}_2 = \frac{6}{ml^2}\frac{8p_{\theta_2} - 3\cos(\theta_1-\theta_2)p_{\theta_1}}{16-9\cos^2(\theta_1-\theta_2)}.$$

The trajectories of the double pendulum for small and large angle cases are shown below. The chaotic phenomena, on the left, can clearly be seen in the large angle case whereas the small angle path, on the right, clearly shows a path that repeats itself over the course of time.

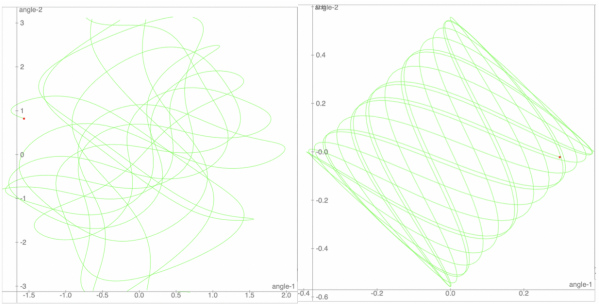

Figure 1: Trajectory of Angles Over Time

In this article, we attempt to model and predict the behaviour of the double pendulum without any prior knowledge about the underlying physics. The input into our various models will be time series data for the first ten to twenty seconds of the system and our output will be the position of the system at future time steps. We will use

the data generated by Runge-Kutta approximation of the equation of motion from first principle.

Related Work

Investigating the abilities of machines to understand the underlying features of data has been an active field of research. A recent paper from ETH Zurich applied an auto encoder network to understand the properties of well known dynamical systems such as planetary orbits and a harmonic oscillator without taking any prior knowledge about the physics of the system into account [1]. Their network performed relatively well and was able to predict the trajectory of the system with high accuracy. The findings from this research raise very interesting implications since it shows that a neural network can identify the underlying pattern and features that govern the behaviour of the phenomena, an ability previously thought to be unique to well-trained physicists. However, not much research has been done in cases where the physical system exhibits non repeated or highly chaotic behaviour.

Objective

We want to determine if we can find a model that can accurately reflect the chaotic motion of a double pendulum given that no prior knowledge about the physics of system is contained within the model. We will try to reflect this motion using the following two methods: 1) training on the first couple seconds of a given initial angle configuration and trying to predict the future motion of the system and 2) trying to model the relationship between successive states in the physical system so that we can determine the entire motion of any given arbitrary initial configuration. By exploring these two avenues we seek to understand the viability of machine learning to understand chaotic systems as a whole.

Dataset and Features

Our dataset contains multiple time series of the states of the double pendulum at different times. The states of the pendulum are described by 8 parameters: the position in x and y coordinates and the velocity in x and y coordinates of the first and second bob $[x_1, x_2, y_1, y_2, v_1^x, v_2^y, v_1^x, v_2^y]$. We obtain the time series data by simulating the dynamics of a double pendulum via the Lagrange equation and Runge-Kutta approximation. In total, we generated 1000 time series each containing 10000 time steps (0.01s interval) for the small angle cases and 1000 time series each contain 20000 time steps for the large angle cases where the chaotic behaviour become more evident. We then process our dataset into two different forms to reflect the two methods talked about in the objective section. For the first method we simply divide each individual time series into a train and test set by splitting it at a certain time point. We will then use these time series to train the regression and LSTM models. For the second method, we make a Markovian assumption that the next state in the double pendulum system is only dependent on the previous state which is reasonable since we know all of the physical parameters that govern the system at any specific point in time. We then pair the states at subsequent times into a tuple $(state_t, state_{t+\delta t})$ where $state_t$ will be the input of the feed forward neural network and $state_{t+\delta t}$ will be the output. There are 1 million pairs of states in total.

Models and Methods

In this section we will provide an overview of all of the algorithms we used in our attempt to model the double pendulum system.

1 Linear Regression With Feature Map

Linear regression is one of the most widely used machine learning algorithms in the case of a regression problem. As known, linear regression takes a hypothesis of the form $\theta^T x$ and seeks to minimize the cost function:

$$J(\theta) = (1/2)\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2$$

As at small angles the pendulum's motion is close to being polynomial, we first seek to apply linear regression with a polynomial feature map to the problem. A polynomial feature map allows for us to take our problem into a higher dimension while still keeping the overall problem a linear one as our hypothesis simply remains $\theta^T \phi(x)$ where ϕ represents our feature map. In the cases we ran, we took polynomial mappings of degree $3, 5, 7$ meaning that one of our feature maps would look like $\phi(x) = [1, x, x^2, x^3]$. As we have that $x \in R^4$, $\phi(x)$ will be the vector of all monomials of x with degree less than 3. We now see that with the feature map included we have that our cost function we seek to optimize will be:

$$J(\theta) = (1/2)\sum_{i=1}^{n}(h_\theta(\phi(x^{(i)})) - y^{(i)})^2$$

2 Autoregressive Model

Auto Regression is a regression predictor that is geared toward time series data. It is commonly used to predict time dependent parameters. In this experiment, we will apply auto regression to predict the future location of the

pendulum based on the information from time 0 to 10s. For simplicity, we will predict only one feature in this step. The Auto Regressive models predict the time series with the following equation (X_t denotes time series data):

$$X_t = c + \sum_{i=1}^{p} \varphi_i B^i X_t + \varepsilon_t$$

We train the auto regression models on the time series of position of upper pendulum from 0 to 8000ms. Then, we will compares the result to the simulated data.

3 Feed Forward Neural Network with Markovian Assumption

The feed forward neural network will be applied to the portion of our dataset where we applied a markovian assumption towards and grouped successive states into pairs. By applying a neural network on a data set of this form we hope that we are able to initialize weights within the parameters of our neural network that reflect the relationship that exists between successive states even though a closed form solution doesn't exist.

In deciding the architecture of our neural network, we decided upon using a ReLU activation function: $f(x) = max(ax+b, 0)$ for the hidden layers and a linear activation function for the output layer as is standard for regression problems. In addition, we took two methodologies behind producing our output: one where we had a single network with an output layer of 8 nodes that produced the next state as a singular vector and the other where we constructed 8 independent neural networks each with an output layer of one node corresponding to a certain feature in the data set. The diagram below describes the two types of methodologies we ran.

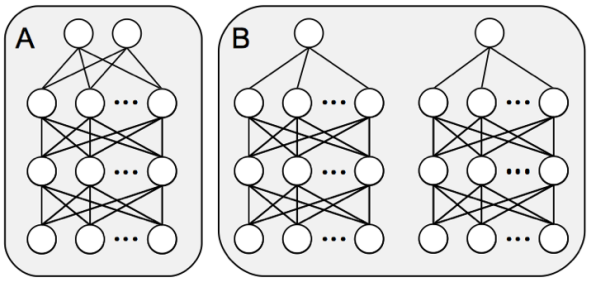

Figure 2: Two NN Frameworks

In generally our hidden layers will contain 8 nodes such that we have one for each of the features of our data. Thus, as we want to learn the weights associated with all of our nodes in the model we see our output equation being:

$$\hat{y} = W^{[num\ layers]} z^{[num\ layers-1]}$$

with loss for the model being:

$$J(y, \hat{y}) = (1/n) \sum_{i=1}^{n} (y^{(i)} - \hat{y^{(i)}})^2$$

where $y^{(i)}$ is the true value.

This means we simply need to take the gradient of this loss with respect to each weight as we backpropagate to update our weights through every run of the data.

4 Long Short-Term Memory Neural Network

LSTM is a variation of Recurrent Neural Network which is generally used with time series and sequential data. It addresses the problem of gradient vanishing that occurs during back propagation through time for vanilla RNN. LSTM can carry information from previous states and is explicitly built to avoid long term dependency problems; therefore, it is ideal for modeling a double pendulum since the next state of the system only depends strongly on a few previous states. We will train the LSTM with different time series of pendulum states in batches from $t = 0$ to $t = 8000ms$. Then we will test the predictions of the model with the data from $t = 8000ms$ to $t = 10000$. Our LSTM has 1 hidden layer with 4 memory states. The architecture of the network is shown below.

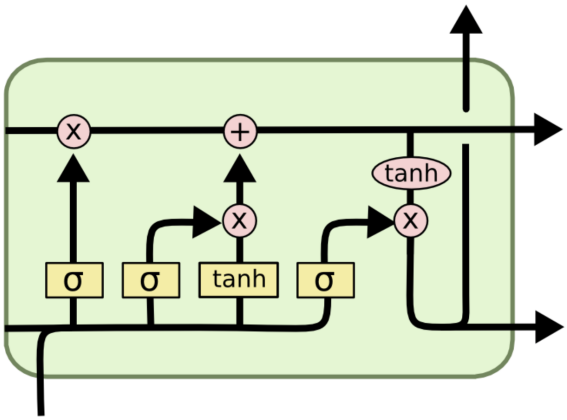

Figure 3: Sketch of LSTM architecture

Where the parameters in the model: C_t, h_t, i_t, f_t and o_t are defined recursively as follows.

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$

$$h_t = o_t \circ \sigma_g(c_t)$$

Results and Discussion

We see that the linear regression model with a 3rd degree polynomial feature map was able to understand the kinematic motion of the double pendulum system at small initial angles fairly well. We ran our model to predict the x position of the first bob given all of the other features along with time. The graph below shows the R2 score for the model on a run over 10 different initial conditions, all of which show that the model was able to explain the variability in the data.

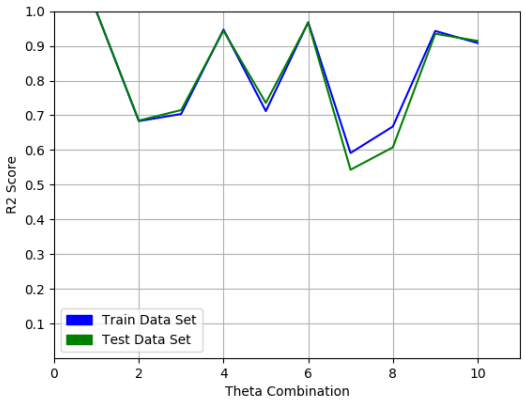

Figure 4: R2 Scores for Linear Regression

We train the autoregression model on the time series of the position of the bob in the upper pendulum from $t = 0$ to $t = 8000ms$ and test it on the simulated data from $t = 8000ms$ to $t = 10000$. The results of the predicted trajectory can be seen in figure 5. The model performs well on the training data (average $MAE = 0.2$); however, the error on the test set is high (average $MAE = 1.1$). This shows that the model overfits the data since autoregression is not complex enough to capture the non-linear relationship of the double pendulum.

Upon training our feed forward neural network we were able to achieve good results on our training data sets. Specifically, we find that the architecture with 2 hidden layers and an output layer of 8 nodes was able to obtain a train MAE of .0679 and a validation MAE of .069. However, testing the network on random initial conditions we see that it was unable to truly understand the chaotic map between states in the double pendulum model. Pictured is the result from predicting the first 20 seconds of a trajectory for a chaotic case.

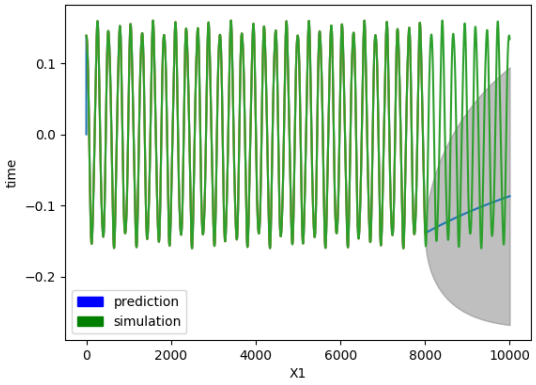

Figure 5: Trajectory Prediction from Autoregression

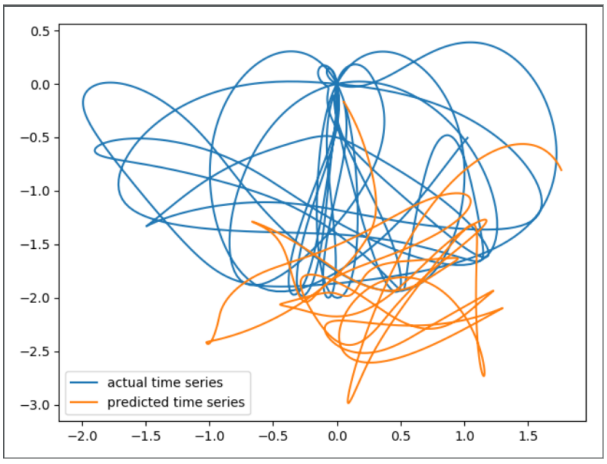

Figure 6: NN Prediction for first 20 seconds of Chaotic Case

The neural network architecture where we created 8 individual networks to learn each feature was not able to capture the relationship between features either. While we see that a network training to predict x_1 was able to obtain an MAE of .00099 on the training set, the model was overfitting for these features and was not able to encapsulate the relationship between all features. Yet, both $L1$ and $L2$ regularization were not able to help improve the model.

We train the LSTM model on time series data ($t = 0ms$ to $t = 8000ms$) across a variety of initial conditions and found that the model was able to predict the trajectory of the pendulum (from $t = 8000ms$ to $t = 10000ms$) very well for both chaotic and non-chaotic cases. We used a batch size of 100 and ran the model for 100 epochs while experimenting with different activation functions before settling on ReLU as it performs the best due to not having a vanishing gradient. Additionally, we found

that normalizing the data with a Min-Max scaler helped the LSTM to converge faster.The average RMSE error is $7.8 * 10^{-3}$ across training data and 0.02 across testing data.

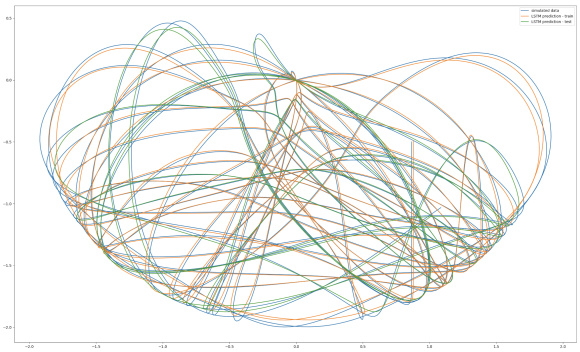

Figure 7: LSTM's predictions vs. simulated data for highly chaotic cases

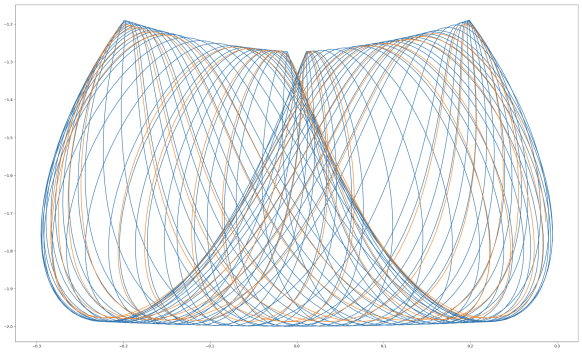

Figure 8: LSTM's predictions (blue line) vs. simulated data (orange line) for periodic case

Error Analysis

We first seek to understand why transforming our data under the Markovian assumption was not able to produce good results. Due to the nature of chaotic systems, determining the relationship that takes one state to the next is very complex and lays in a large phase space. This results in our neural network being unable to capture all of the details within the mapping between two states of the system and having high variance. Additionally, we see that even if our model was able to produce a close approximation of the map, having even small variability in the output state will lead to drastically different trajectories in the chaotic system leading to an inaccurate model.

In relation to the LSTM model we see that normalizing the data was very helpful. As the nature of the system produces wild results that fluctuate across a range of values, scaling the data to be within the values of 0 and 1 allowed for the model to gain a more accurate understanding of the system. Then, by inverting this scaling as the model output it's results we see that the projected trajectory was highly accurate. In addition we see that this normalization is able to reduce variance in the model.

Conclusion and Future Work

For small angle cases autoregression and linear regression were able to capture the pattern of the pendulum relatively well. For large angle cases, running the LSTM network on normalized data was able to predict the chaotic nature of the double pendulum with high accuracy given previous time series data for the specific initial condition. However, the ANN model had high error in its prediction of the trajectory of the system given only an initial state as the process is very chaotic and the data space is extremely large. Due to the inability of the ANN to predict the path of an arbitrary initial angle we see that a different neural network architecture might be needed to capture the subtle details of the system.

Future work will explore ways to generalize the ANN network to learn the chaotic map between two consecutive states in the pendulum so that it can predict the entire path of the double pendulum for any given initial condition. Experimentation with different LSTM architectures to incorporate the multivariate features of the data is another possible avenue of research. Ultimately, we would like to build a model that captures all the physical relationships in the system which will guide us to a better understanding of chaotic system.

The code for our project is available at the following link. `https://github.com/jparmar32/CS229-Final-Project-`

Contributions

Jupinder	Implementation of Linear Regression and Feed Forward Neural Network, Error Analysis, Report Writing
Sirapop	Data Collection and Preprocessing, Implementation of Autogregressive Model and LSTM, Report Writing

Figure 9: Contributions

References

1. Raban, Iten et al. 2018. *Discovering physical concepts with neural networks*. arXiv

2. Pathak, Jaideep et al. 2017. *Using Machine Learning to Replicate Chaotic Attractors and Calculate Lyapunov Exponents from Data*. arXiv

3. Basharat, Arslan et al. 2017. *Time Series Prediction by Chaotic Modeling of Nonlinear Dynamical Systems*. arXiv

4. Wu, Tailin et al. 2019. *Toward an AI Physicist for Unsupervised Learning*. arXiv

5. OmkarThawakar. 2018. *Forecasting-Trajectory-of-Double-Pendulum*. Github

6. Singh, Aishwarya. 2018 *A Multivariate Time Series Guide to Forecasting and Modeling (with Python codes)*.

7. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. and Kudlur, M., 2016, November. *Tensorflow: a system for large-scale machine learning*. In OSDI (Vol. 16, pp. 265-283).

8. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J., 2011. *Scikit-learn: Machine learning in Python*. Journal of machine learning research, 12(Oct), pp.2825-2830.

9. Olah, C. *Understanding LSTM Network* 2015

10. Guerrini, T. *Neural Networks For Multivariate Regression* 2017