Solving differential equations: Double pendulum and deep learning

Bálint Hantos (DVTULF)

Introduction

The aim of my project is to investigate how neural networks perform at solving ordinary differential equations. Currently, machine learning solutions are trending in the IT industry. From image and sound processing methods through recommender systems to time series analysis it is getting widespread to use neural networks in fields where large amounts of data is available.

Motivation

The motivation behind this project stems from the strong intertwining between differential equation solving and the nature of neural networks. It is very important to mention, that there is a lot of related work in this area. Notably, recently solvers were used to suggest some new neural network architectures [1,2] as well as new training methods [3].

The usage of neural networks to tackle physical problems regarding ODE solving isn't very new. My personal motivation of choosing this project conceived upon reading an article about using the MLP-model to solve the chaotic three-body problem [4]. Firstly, I thought that the model presented in the article was poorly developed, due to the fact that the MLP-model can't take successivity into account.

On the other hand, recurrent neural networks, as a general tool of time series analysis, can be of a good use, to tackle the notion of events being sequential. I propose a model in this project, which might be capable of solving the equation of motion for the chaotic double pendulum.

Main ideas and basic equations

The double pendulum is one of the first encounters with analytically unsolvable physical systems during undergraduate physics. Although, its differential equations can be solved iteratively, usually on a computer. Also, it is a good example of a chaotic system.

During undergrad, we mostly used the Runge-Kutta-method with fixed and adaptive step sizes, and we compared it to the Euler-method and the Euler-Cromer-method (in the case systems of differential equations).

The double pendulum is a pendulum with another pendulum attached to its end. It exhibits rich dynamic behavior and it's strongly sensitive to its initial conditions. The two pendula can be described by the masses (m_1, m_2) hanging on them, and their l_1, l_2 lengths. The generalized coordinates can be conveniently chosen to be θ_1, θ_2 angles between each limb and the vertical.

The Lagrangian of such system is:

$$L = \frac{m_1 l_1^2}{2}\dot{\theta_1^2} + m_1 g\, l_1\, cos\ \theta_1 + \frac{m_2}{2}\cdot\left(l_1^2\dot{\theta_1^2} + l_2^2\dot{\theta_2^2} + 2l_1 l_2\dot{\theta_1}\dot{\theta_2}\cos(\theta_1-\theta_2)\right) +$$

$$+ m_2 g\cdot(l\cos\theta_1 + l_2\cos\theta_2).$$

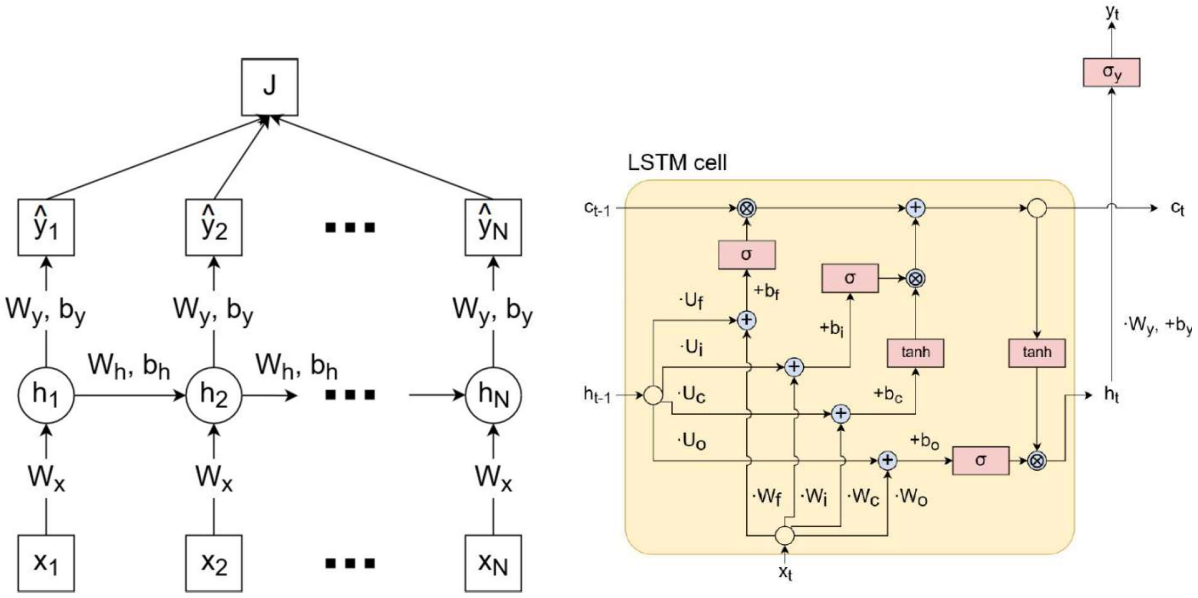
Thus the equations of motion:

$$\ddot{\theta_1} = \frac{-g(2m_1+m_2)\sin\theta_1 - m_2 g\sin(\theta_1-2\theta_2)}{l_1(2m_1+m_2-m_2\cos(2\theta_1-2\theta_2))} - \frac{2m_2\sin(\theta_1-\theta_2)\left(\dot{\theta_2^2}l_2 + \dot{\theta_1^2}l_1\cos(\theta_1-\theta_2)\right)}{l_1(2m_1+m_2-m_2\cos(2\theta_1-2\theta_2))},$$

$$\ddot{\theta_2} = \frac{2\sin(\theta_1-\theta_2)\left(\dot{\theta_1}^2 l_1(m_1+m_2)\right)}{l_2(2m_1+m_2-m_2\cos(2\theta_1-2\theta_2))} + \frac{2\sin(\theta_1-\theta_2)\,g(m_1+m_2)\cos\theta_1}{l_2(2m_1+m_2-m_2\cos(2\theta_1-2\theta_2))} +$$

$$+ \frac{2\sin(\theta_1-\theta_2)\,\dot{\theta_2^2}l_2 m_2 cos(\theta_1-\theta_2)}{l_2(2m_1+m_2-m_2\cos(2\theta_1-2\theta_2))}.$$

The MLP-model in the [3] paper can only work with fixed size input and output. We need to use a different approach. Instead of processing the whole series of data in one step, we only look at two elements – the current and the previous element – of the series in each step. Therefore, it is necessary for the neural network to store the relevant information about the previous states. For this task a recurrent neural network is a good choice.



1. Figure: VNN (left) and LSTM (right) architectures [5]

I intend to examine how the vanilla recurrent neural network (VRNN) and the long short-term memory (LSTM) architecture perform in this task. Firstly, I'm going to use the Runge-Kutta ODE solver to generate datapoints of a motion of a double pendulum with the same parameters (m_i, l_i) and different initial conditions. Secondly, train the above architectures on the generated data to predict the trajectory. Finally, measure the goodness of the fit of the trajectory with mean squared error and energy conservation.

The tools to implement such networks can be found in the `Keras` open-source neural network library. It is written in `Python` and can run on top of `TensorFlow`.

References

[1] Haber, Eldad, and Lars Ruthotto. "Stable architectures for deep neural networks." Inverse Problems 34.1 (2017): 014004.

[2] Chen, Ricky TQ, et al. "Neural ordinary differential equations." *Advances in neural information processing systems*. 2018.

[3] Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, and Elliot Holtham. Reversible architectures for arbitrarily deep residual neural networks. arXiv preprint arXiv:1709.03698, 2017.

[4] Breen at al.: Newton vs the machine: solving the chaotic three-body problem using deep neural networks

[5] Viktor Varga: Deep neural networks: algorithms and architectures course at ELTE Department of Informatics