Learning about Neural ODE

Tyrone DeSilva

AMATH University of Washington

Motivation

- Understand how adjoint methods can be used to back-propagate gradients through a black-box ODE solver
- Explore differentiable programming
- Time series interpolation

Background

Neural Ordinary Differential Equations(NODE) are a kind of continuous depth neural network.

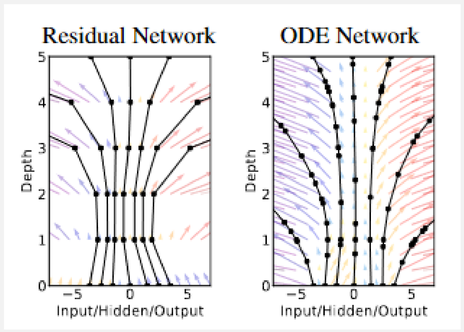

Figure: NODE is inspired in part by ResNET [1].

$$\frac{dh(t)}{dt} = f(h(t), t, \theta)$$

f is a time-invariant neural network. The main contribution of [1] is to apply adjoint sensitivity method to make backpropagating through an ODE solution more efficient and treat solver as a black box.

- Control numerical error
- Linear with problem size
- Solves backward in a single pass of an augmented ODE problem.
- Low memory cost.

Backpropagation Through RK4 Solver

$$y_{n+1} = y_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4)$$
$$k_1 = f(t_n, y_n, \theta)$$
$$k_2 = f(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}, \theta)$$
$$k_3 = f(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}, \theta)$$
$$k_4 = f(t_n + h, y_n + hk_3, \theta)$$
$$\frac{dy_n}{d\theta} = \frac{\partial y_n}{\partial \theta} + \frac{\partial y_n}{\partial y_{n-1}}\frac{dy_{n-1}}{d\theta}$$
$$\frac{d\mathscr{L}(y_1, y_2, \ldots, y_N)}{d\theta} = \sum_{i=1}^{N} \frac{\partial \mathscr{L}}{\partial y_i}\frac{dy_i}{d\theta}$$

Fitting a NN to LKV Data

In this case $f(x_1, x_2)$ is a FC NN which maps from $\mathbb{R}^2$ to $\mathbb{R}^2$. Even if we don't know that the system is LKV, we can apply this method of backpropagating through the RK4 solver.

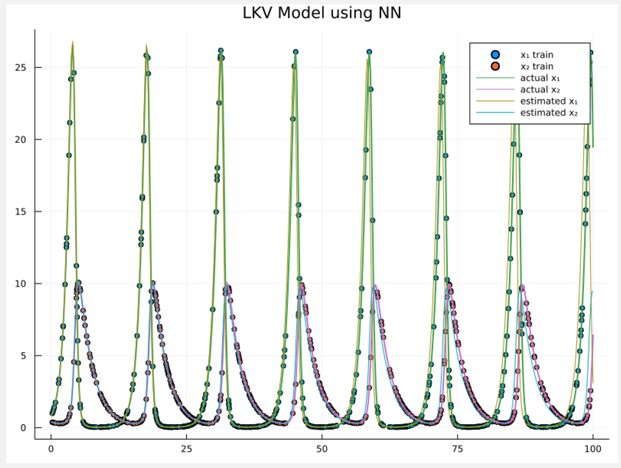

Figure: Fitting NN to LKV Data

Generating LKV Data

Lotka-Volterra Model

$$\frac{dx_1}{dt} = \alpha x_1 - \beta x_1 x_2,$$
$$\frac{dx_2}{dt} = \delta x_1 x_2 - \gamma x_2$$

N points are sampled uniformly across the time-span, giving us an irregularly sampled time series.

Fitting LKV Model Parameters

If we know the data generating process is a LKV model, we can apply this method. In this case, we can just use

$$f(x_1, x_2) = \begin{pmatrix} \frac{dx_1(x_1,x_2)}{dt} \\ \frac{dx_2(x_1,x_2)}{dt} \end{pmatrix}$$

and learn the parameters $\alpha, \beta, \delta, \gamma$ using gradient descent.

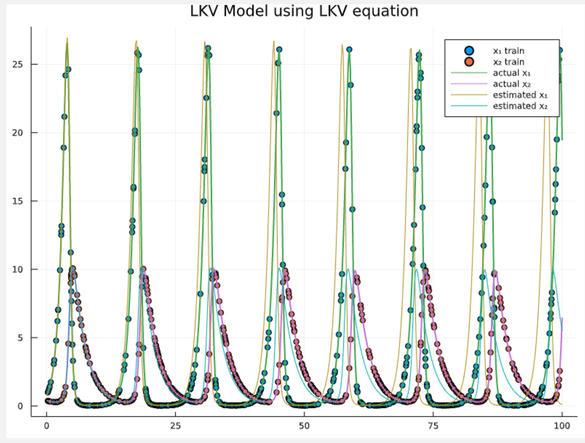

Figure: Fitting LKV Model Parameters

True and learned parameters:

$$(\alpha, \beta, \delta, \gamma) = (1.1, 0.4, 0.1, 0.4)$$
$$(\hat{\alpha}, \hat{\beta}, \hat{\delta}, \hat{\gamma}) = (1.096, 0.399, 0.101, 0.419)$$

Learning LKV Using NODE

Similar to the case of fitting a NN to the LKV data through the RK4 solver, but this time we backpropagate using the adjoint sensitivity method. This works for any ODE solver.

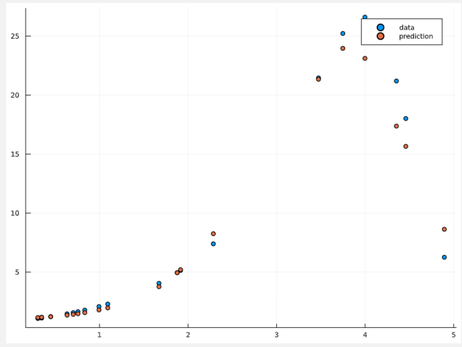

Figure: NODE Predictions for Small Subset of LKV Data

Next Steps

- Latent ODE
- Train NODE to fit entire LKV
- Try fitting other dynamical systems
- Implement adjoint sensitivity
- Video Interpolation [2]

References

[1] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud.
Neural ordinary differential equations.
CoRR, abs/1806.07366, 2018.

[2] Sunghyun Park, Kangyeol Kim, Junsoo Lee, Jaegul Choo, Joonseok Lee, Sookyung Kim, and Edward Choi.
Vid-ode: Continuous-time video generation with neural ordinary differential equation, 2021.

Contact Information

- Github: tqhdesilva/AMATH563-final-project
- Email: tdslv@uw.edu