luisf.canals@gmail.com, 2023

Introduction

First consideration is the Universal Approximation Theorem, which establish in general, that a standard multilayer feed-forward neural network with just only a single hidden layer, with a finite number of neurons, can approximate continuous function in a compact subset of the space of real vectors.

Initially, if we think about the logistic discrete equation:

$$x(n+1) = r * x(n) * (1-x(n))$$

And we try for r between 3.6 and 4 to predict the value of x(n) given x(0) and n.

We might do it just iterating the equation, or we can try to create and train a ML with thousands of cases and try to apply to any x(n).

Let's focus on the case r=4. For that value, Schröder found an analytic solution:

$$x(n) = \sin^2 (\sin^2(x(0)) * 2^n)$$

We want to use the logistic function in, for instance, the interval of n=0,1,2,….. 10^20 and x(0) between 0 and 1 including both. From a mathematical point of view, the subset {0,1,2,3,…10^20} x [0,1] into R x R is closed and bounded, so it's compact.

And obviously, Schröder solution is continuous (even differentiable) in R x R.

According to Universal Approximation Theorem, we can approximate this function with a RNN (the core of ML). Nice!

We can think if a model able to approximate sin/cos functions might approximate Schröders solution.

To do that, I've implemented a LSTM model that works really well for sin(x), cos(x), sin(x^2)

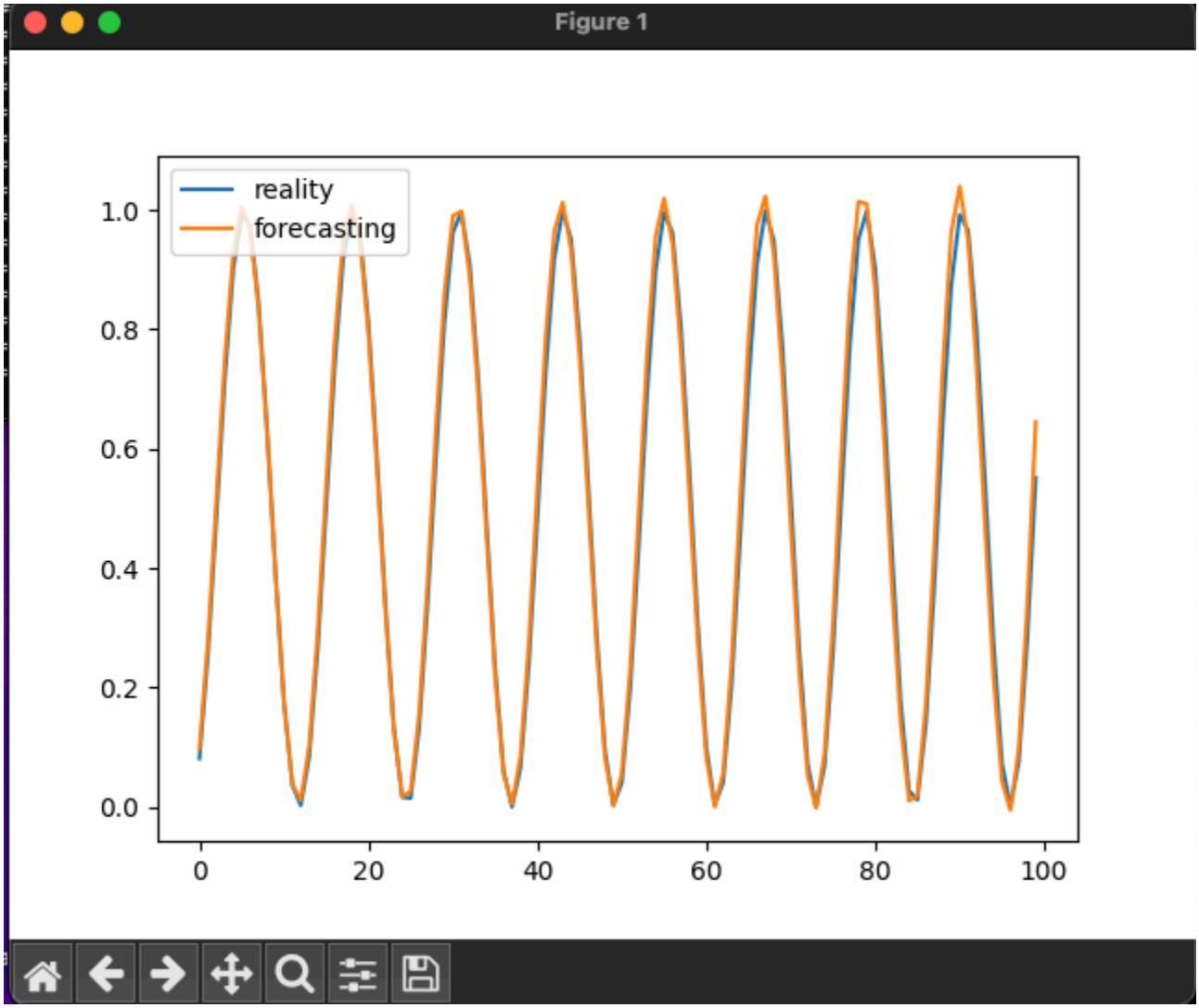
What is expected from the chaotic functions, as they have sensibility to initial conditions. The calculated sequence might vary a lot due to numerical instabilities.

For the starting point I'm going to consider two choices:

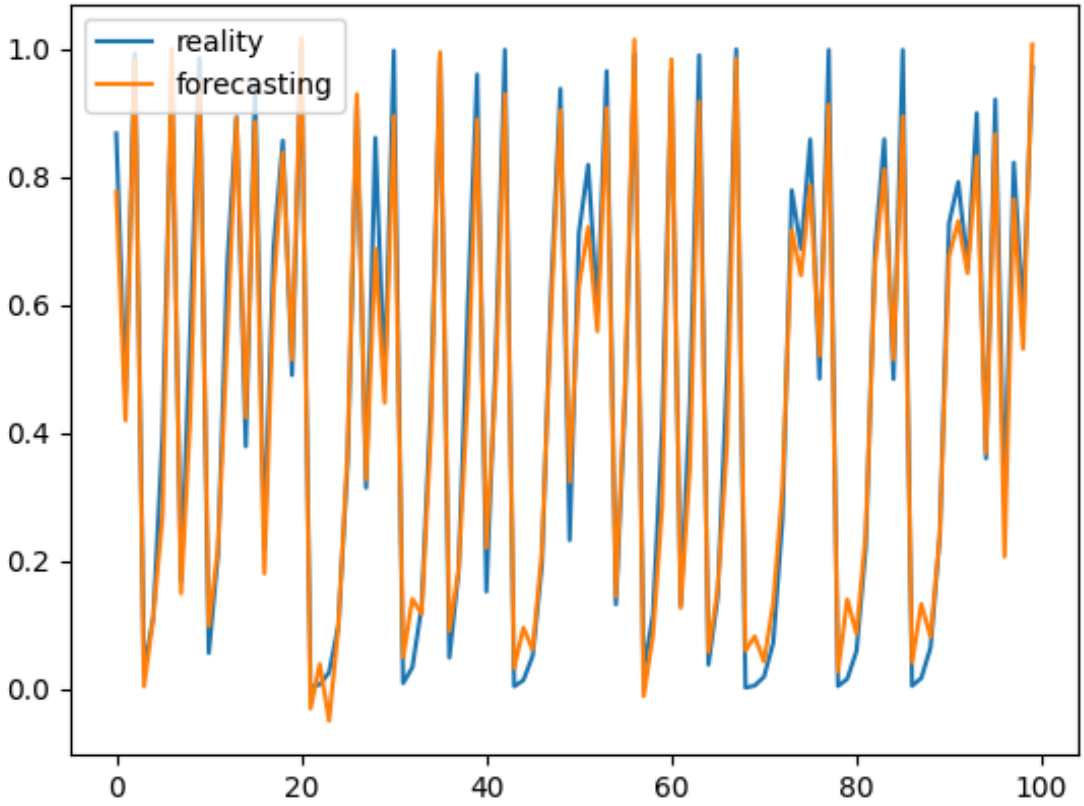
- With 32 neurons and 2 layers the fit is almost perfect for sin (and the training time is 115s)
- With 4 neurons and 2 layers the fit is almost perfect also (training time 109s)

Some experimental results

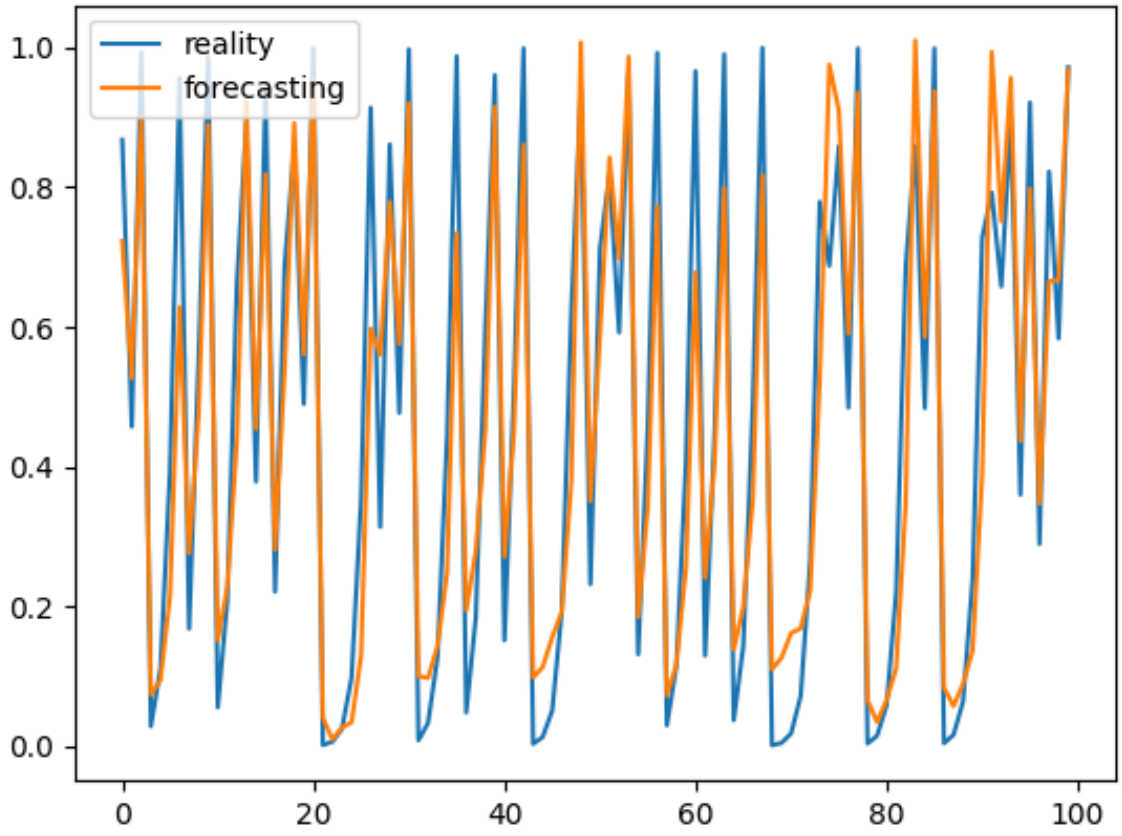
F
or sin(x^2) functions I used 2 layers of 32 neurons. The adjust is almost perfect:



Trying to forecast Schrödingers analytic solution to logistic discrete equation (R=4) and x0=0.1234 using the same schema as for the previous cases, the results are not really bad:

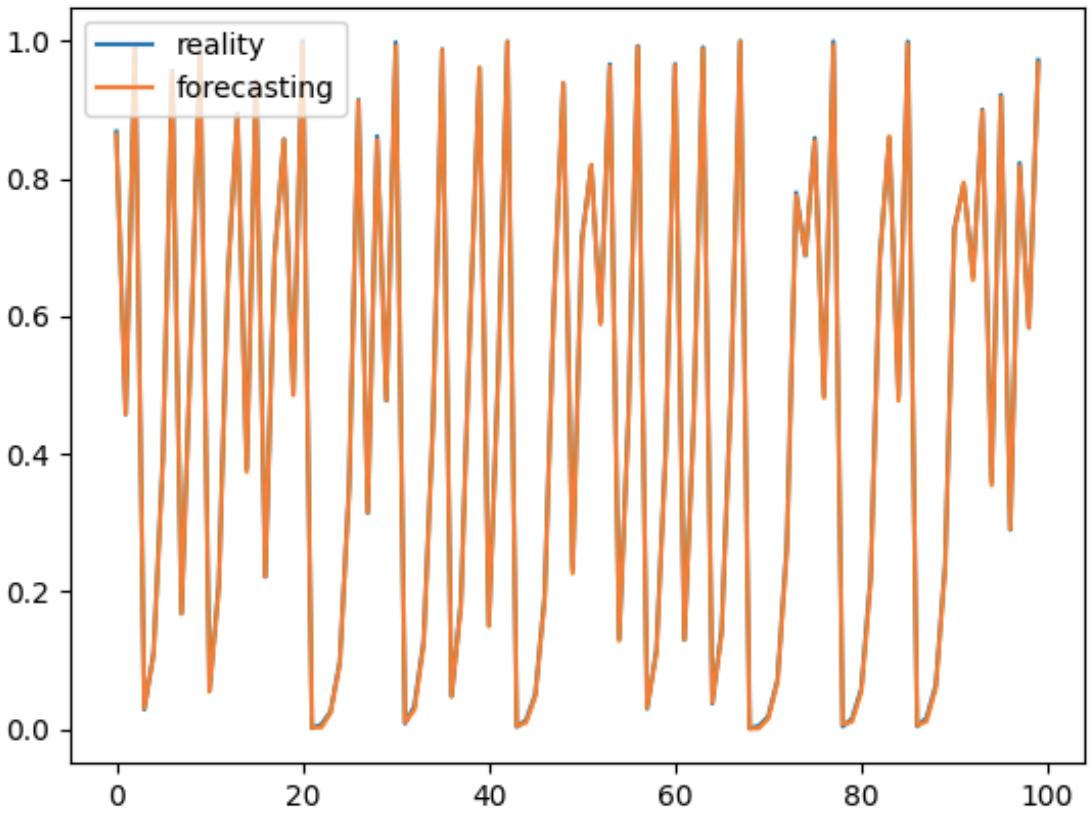


We can try again with more layers and neurons (4 layer and 128 neurons):



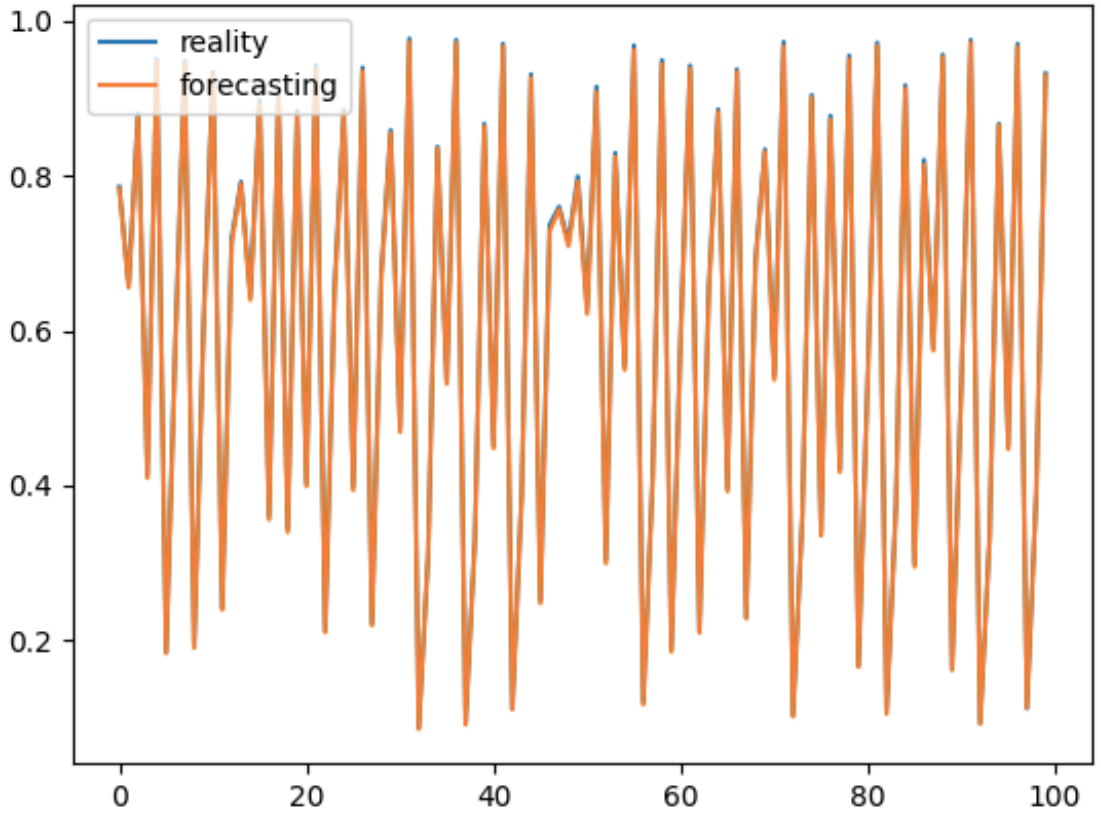
What I've observed it's not a question of having more neurons and layers but more train attempts. With 1000 epochs, the error loss has the opportunity to decrease to 0.001, that guarantees a good approach. With an error during training of around 0.1, there will be no valid forecasting…

The result with 1000 epochs and only 32 neurons in 2 layers:



With only 32 neurons, 2 layers, but 1.000 epochs of training, we are able to fit perfectly the chaotic logistic equation

$$x(n+1) = 3.92 * x(n) * (1-x(n))$$
$$x(0) = 0.1234$$



References

Universal Approximation Theory: https://arxiv.org/pdf/1910.03344.pdf

Discrete logistic eq: https://francis.naukas.com/2020/11/28/carnaval-de-matematicas-la-solucion-exacta-de-la-ecuacion-logistica-y-el-caos-como-periodo-irracional/

Approximation to sin with RNN: https://stackoverflow.com/questions/13897316/approximating-the-sine-function-with-a-neural-network

https://www.sciencedirect.com/science/article/abs/pii/096007799580045I. “Learning chaotic dynamics by neural networks

https://medium.com/@krzysztofbalka/training-keras-lstm-to-generate-sine-function-2e3c0ca42c3b

https://www.sciencedirect.com/science/article/abs/pii/S0960077998003026

https://www.sciencedirect.com/science/article/pii/S096007792100924 3#bib0027

https://machinelearningmastery.com/time-series-prediction-with-deep-learning-in-python-with-keras/

https://datascience.stackexchange.com/questions/19365/predict-sinus-with-keras-feed-forward-neural-network