# PINN, MSNN and Reservoir Computing

Zhen ZHANG

August 8, 2019

# Table of Contents

# Problem Setup

$$\frac{dx}{dt} = \sigma(y - x)$$
$$\frac{dy}{dt} = x(\rho - z)$$
$$\frac{dz}{dt} = xy - \beta z$$

- For $\sigma = 10$, $\rho = 28$, $\beta = \frac{8}{3}$, the solution to the system above is chaotic with largest Lyapunov exponent around 0.91.
- For our test case, we consider initial condition $x(0) = -8$, $y(0) = 7$, $z(0) = 27$.
- We use training data from $t = 0$ to $t = 100$ with $\Delta t = 0.02$ to predict the solution from $t = 100$ to $t = 125$.

# Multistep Neural Network

- Assume $\frac{d}{dt}\mathbf{x}(t) = f(\mathbf{x}(t))$, f unknown. Give value of $\mathbf{x}$ from $t = 0$ to $t = t_0$ (discrete) as training data.
- Place Neural Network prior on f: $f \sim NN(x)$
- Try to minimize value $L = \frac{d}{dt}\mathbf{x}(t) - f(\mathbf{x}(t))$
- Challenge: Only have $x(t)$ as discrete points, not a continous function, thus cannot take derivative
- Remedy: Use Linear Multistep difference operator as a replacement to the differential operator, i.e., minimize

$$L = \Sigma_{n=M}^{N} |y_n^2|,\ y_n = \Sigma_{m=0}^{M} [\alpha_m x_{n-m} + \Delta t \beta_m f(x_{n-m})]$$

- Once $f$ is learned, predict the solution of $\mathbf{x}$ using traditional ODE solver.

# Multistep Neural Network

- Neural Network architecture: 3 hidden layer with 256 hidden units, *tanh* activation function.
- Variation: Place a different prior $f \sim Resnet(x)$
- A resnet is a composite of residual blocks, denoted by $R$. $R(x) = \sigma(W_2\sigma(W_1(x))) + x$.
- One fully connected layer, followed by a residual block and an output layer, each with 256 units.

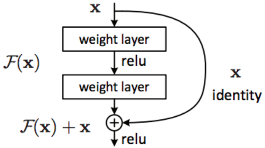


Figure: Architecture of a Residual Block

# Reservoir Computing

- Reservoir Computing is a special form of RNN.
- RNN model: Input $x(t)$ from $t = 0$ to $t = t_0$.

$$r(t) = \tanh(W_1 r(t-1) + W_2 x(t-1))$$

$$\tilde{x}(t) = W_{out} r(t), r(0) = 0$$

  Training: minimize $L = ||x(t) - \tilde{x}(t)||$ to get $W_1$, $W_2$ and $W_{out}$.
  Prediction: Given $x(t_0)$, can get $\tilde{x}(t_0)$, then replace $x$ by $\tilde{x}$ to get $\tilde{x}(t_0+1)$, $\tilde{x}(t_0+2)$...
- Reservoir Computing: Fix $W_1$ and $W_2$, only train $W_{out}$.
- Advantage: Fast and easy to implement: just linear regression.
- Details: $W_1$ is the adjacency matrix of a sparse random graph, elements of $W_2$ are uniformly distributed.

# PINN

- No need to introduce
- Prior: $x \sim NN(t)$, compared to $f \sim NN(x)$ in MSNN
- Here, we solve the forward problem without using data, but assume knowing the equation.
- So far, no way of solving the equation when system is chaotic ($\rho = 28$).
- Can work when system is not chaotic ($\rho = 20$)

# Feed Forward vs ResNet

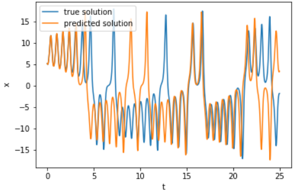


Figure: $x(t)$, $t \in [100, 125]$, using feed forward NN

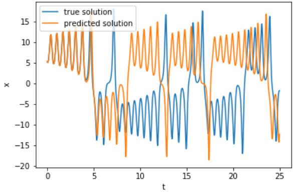


Figure: $x(t)$, $t \in [100, 125]$, using Resnet

# Feed Forward vs ResNet

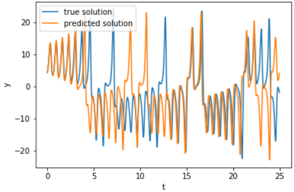


Figure: $y(t)$, $t \in [100, 125]$, using feed forward NN

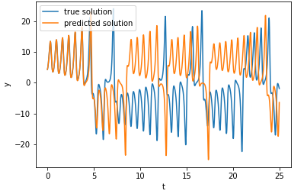


Figure: $y(t)$, $t \in [100, 125]$, using Resnet

# Feed Forward vs ResNet

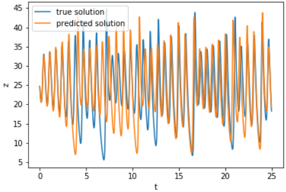


Figure: $z(t)$, $t \in [100, 125]$, using feed forward NN

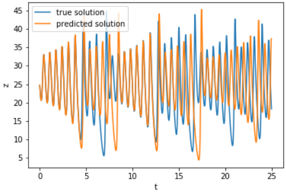


Figure: $z(t)$, $t \in [100, 125]$, using Resnet

# RC Result

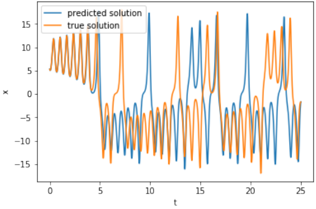


Figure: $x(t)$, $t \in [100, 125]$, using RC

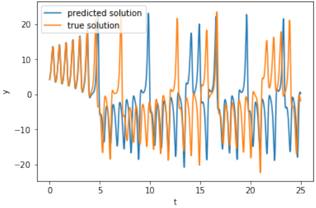


Figure: $y(t)$, $t \in [100, 125]$, using RC

# RC Result

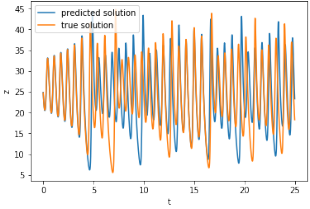


Figure: $z(t)$, $t \in [100, 125]$, using RC

# Feed Forward vs ResNet vs Reservoir vs PINN

Table: Comparison between RC and MSNN on solving Lorenz 63

| | Parameters | Assumption | VPT |
|---|---|---|---|
| RC | 91800 | No | 3.66 |
| MSNN | 132608 | $u' = f(u)$ | 4.29 |
| ResNet | 132608 | $u' = f(u)$ | 5.36 |
| PINN | ?? | $u' = f(u)$, f known | ?? |

$$\mathrm{VPT} = \mathrm{argmax}_{t_f}\{t_f \mid \mathrm{NRMSE}(\mathbf{x}(t)) < \epsilon, \forall t \le t_f\}$$

$$\mathrm{NRMSE}(\mathbf{x}) = \sqrt{< \frac{(\mathbf{x} - \tilde{x})^2}{\sigma^2} >}$$

- Because of randomness of the networks, we train each network for 10 time and take the average result.

# Table of Contents

# Problem

- Given data points $\mathbf{x}(t)$, from $t = 0$ to $t = t_0$, can we make prediction using PINN, i.e., predict what is happening after $t = t_0$. Here, we don't know the parameters $\rho$, $\sigma$, $\beta$.
- We can use backward PINN to learn the parameters. But 2 problems are encountered:
- First, backward PINN doesn't always give a correct result, especially when the data points are sparse. We can overcome this by changing the neural net architecture to LSTM. We put error table in next page.
- Second, after we learn the parameters, normally with some small error, the traditional DE solver cannot give long time prediction, when the system is chaotic.

# Problem

Table: Comparison between FNN and LSTM on inferring Lorenz 63

| | $\sigma$ | $\rho$ | $\beta$ | Rel Err |
|---|---|---|---|---|
| FNN | 15.636 | 26.641 | 1.538 | 34.51% |
| LSTM | 9.983 | 27.942 | 2.652 | 0.31% |
| True | 10 | 28 | $\frac{8}{3}$ | |

# Problem

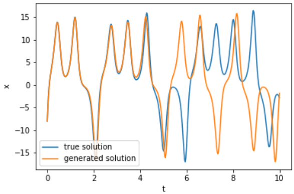


Figure: $x(t)$, $t \in [0, 10]$

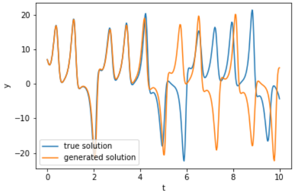


Figure: $y(t)$, $t \in [0, 10]$

# Hybrid Model

- Combine a good short-term DE solver with Reservoir Computing
- Hybrid model: Input $x(t)$ from $t = 0$ to $t = t_0$.

$$r(t) = \tanh(W_1 r(t-1) + W_2 x(t-1) + W_3 K[x(t-1)])$$

$$\tilde{x}(t) = W_{out} r(t) + \tilde{W}_{out} K[x(t-1)], r(0) = 0$$

- Compare to original model:

$$r(t) = \tanh(W_1 r(t-1) + W_2 x(t-1))$$

$$\tilde{x}(t) = W_{out} r(t), r(0) = 0$$

- Can expect better predictions.

# Hybrid Model

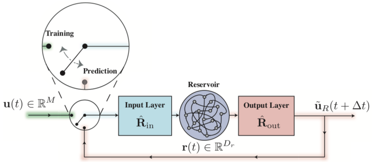


Figure: Reservoir Only Model

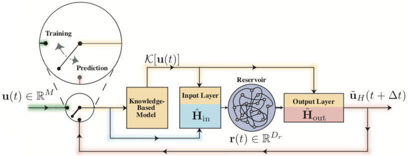


Figure: Hybrid Model

# Hybrid Model

- Assume in Lorenz system, $\rho$ is changed by 0.05, we can generate long prediction up to $t = 12$.
- I cannot recover Ott's result, possibly because they use some tricky initialization of matrices.
- But it can be imagined that the adapted PINN can work well in this case.

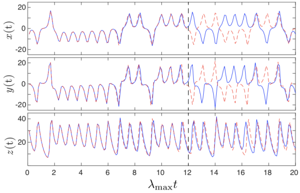


Figure: Prediction result using Hybrid Model

# Table of Contents

# Conclusion

- Multistep Neural Network works better compared to RC given similar parameter space(?).
- Multistep Neural Network works better when using a Resnet architecture.
- PINN and RC can be combined to generate a hybrid model, which can improve prediction accuracy.

# Future Work

- Tune hyperparameters to generate a good result combining PINN and RC
- Test the result on Mathieu Equation
- Make PINN work on forward problem of Lorenz equation
- Generate more systematic result on a wide range of parameters by varying $\rho$ (subsequently varying Lyapunov exponent)
- Try long term integration using Parareal and transfer learning.
- ...

# The End