Differential Informed Auto-Encoder

Zhang Jinrui*
jerryzhang40@gmail.com

20241021

Abstract

In this article, an encoder was trained to obtain the inner structure of the original data by obtain a differential equations. A decoder was trained to resample the original data domain, to generate new data that obey the differential structure of the original data using the physics-informed neural network [4, PINN].

1 Introduction

If the physics formula was obtained in the form of differential equations, a physics-informed neural network can be built to solve it numerically on a global scale [4, PINN].This process could be seen as a decoder in a way that takes a sample point in the domain of the partial differential equations, and solve it to get the corresponding output of each input point. If only a small and random amount of training data was obtained, to re-sample from the domain, we need to obtain the differential relationship of the data. This process could be viewed as an encoder that encodes the inner structure of the original data. And the decoder decode it by solving the differential equations.

2 Methodology

2.1 first approach

The first idea is simple. For a one-variable function $u(t)$, define a second-order differential equation in its general form $(\forall t)(F(\frac{d^2u}{du^2}, \frac{du}{dt}, u) = 0)$.

The data of the function $u(t)$ are given in tuples denote as $(T, U)_i \equiv (T_i, U_i)$. And it is natural to denote the differentials by U_i^t and U_i^{tt}. There are several methods to compute these two differentials, including just using the definition of the derivative. In this article, local PCA are compute to obtain these differentials. Local PCA means finding the nearest K neighbors of a given point, which K is a hyper parameter, and performing PCA on these points close to each other

*Some other guy waiting for add

to get the principal direction. The slope of this direction is the derivative U^t in general. Repeat this process on (T, U^t) to obtain U^{tt}

Create a FCN denote as f to represent $F(\frac{d^2u}{du^2}, \frac{du}{dt}, u)$ F to be 0 at every data points and to be 1 all elsewhere is wanted.

To achieve these requirements, we evaluate f at all the data points, and train the network to evaluate these points to 0. Then randomly sample the points of $\mathbb{R}^3$ and train these points to be 1 Algorithm1.

Algorithm 1 f trainer

Require: Input parameters $f, T_i, U_i, U_i^t, U_i^{tt}$

Initialize f randomly
repeat
  $F_i \leftarrow f(U_i, U_i^t, U_i^{tt})$
  $RAND_i \leftarrow$ randomly sample in $\mathbb{R}^3$
  $R_i \leftarrow f(RAND_i)$
  $L \leftarrow meanSquareError(F_i, 0) + 0.1 * meanSquareError(R_i, 1)$
  backPropagation against L to optimize f
until L meets requirement
return f

Once The f was obtained, we can perform PINN as a decoder to generate new data.

The experiment code for the pictures in Results can be run by a Python program in Github [1, deSineTasks] The requirement environment may be installed using [2, reqs]

2.2 approach with linear assumption

For a small randomly sampled data set, the data points in the vector space of differentials $\begin{bmatrix} U & U^t & U^{tt} \end{bmatrix}$ are very low dimensional manifold which in the second order differential equation case, the manifold have to be a two dimensional manifold to satisfy it is one equation. As the pictures show below Figure1, if we only have some of the experimental data input, with the algorithm in the first approach, we would set all the value in the one dimensional manifold to 0 but all elsewhere to 0 which is only one specify solution with the same initial condition as the input data. To have more generalization ability, sacrifice are necessarily taken with some of the flexibility to be able to learn all kind of weird data structures, but assume we are learn a Linear equation at the first place. In this specific case, the ring shape data points in Figure1 need to be treat as a plane cross through the rings span.

As long as the data are on the spanned plane. we will have a same differential structure as the data set. With this assumption, basically only a specific clean data set are required for only one initial condition.

To achieve this result also very straight forward. Perform all the method, that has been used to compute the differential vector $\begin{bmatrix} U & U^t & U^{tt} \end{bmatrix}$ in the first

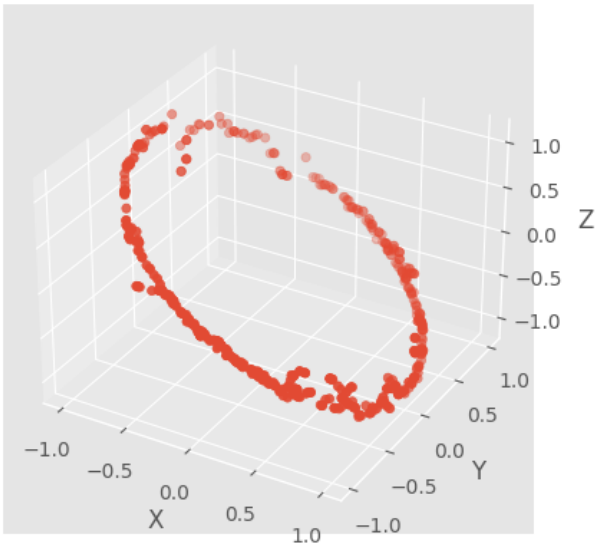

Figure 1: vector $\begin{bmatrix} U & U^t & U^{tt} \end{bmatrix}$ sit on 1-dimensional manifold

approach. Either directly get the derivatives or perform other methods.

After U_i, U_i^t, U_i^{tt} are obtained, the latent space of the differential relationships are obtained. With the Linear assumption, the equation is Linear so all the valid points sit on one same plane. Directly perform PCA on these data points, then take the eigen vector v of the smallest singular value, the direction that need to be reduced has been found. Simply define the encoder function as $f(x) = v \cdot x$ Algorithm2.

Then minimizing f will reduce one dimension Linearly from the latent differentials space. Which can view as f has done the job originally constrained by the differential equation.

This method can find all the Linear differential relationship i.e. the Linear differential equations, from a single function from the entire solution function family. The experiment result can be obtained by Github code [3, deLinearTasksSine]

2.3 approach augmentation

This method is a autoencoder we can also check the difference betwen the input data and the data generate by the PINN in the differential relationship

Algorithm 2 f trainer

Require: Input parameters $f, T_i, U_i, U_i^t, U_i^{tt}$

Initialize f randomly
repeat
  $F_i \leftarrow f(U_i, U_i^t, U_i^{tt})$
  $RAND_i \leftarrow$ randomly sample in $\mathbb{R}^3$
  $R_i \leftarrow f(RAND_i)$
  $L \leftarrow meanSquareError(F_i, 0) + 0.1 * meanSquareError(R_i, 1)$
  backPropagation against L to optimize f
until L meets requirement
return f

constrain. This can view as a parameterize method.

3 Results

3.1 first approach

Train the model on a pure $sin(x)$ and try to get a result meet the initial condition with $U_0^t = 0.5$ and $U_0 = 0.0$ which the exact solution is $0.5 * sin(x)$ would be required output. Result shows in Figure2.

3.2 approach with linear assumption

Train the model on a pure $sin(x)$ and try to get a result meet the initial condition with $U_0^t = 0.5$ and $U_0 = 0.5$ which the exact solution is $\frac{\sqrt{2}}{2} * sin(x + \frac{\pi}{4})$ would be required output. Result shows in Figure4.

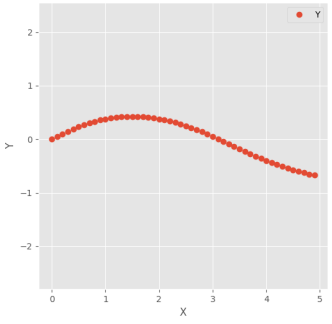

Figure 2: $0.5 * sin(x)$

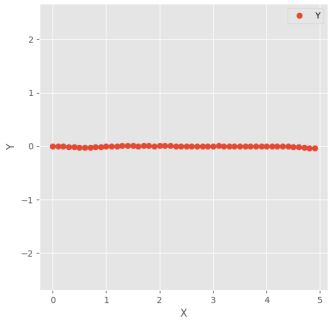

Figure 3: f errors

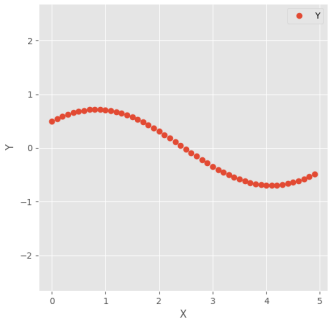

Figure 4: $\frac{\sqrt{2}}{2} * sin(x + \frac{\pi}{4})$

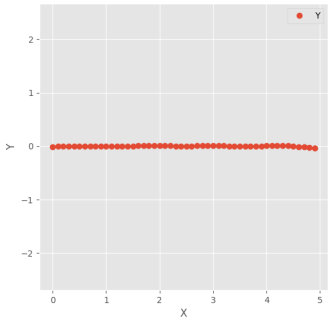

Figure 5: f errors

4 Conclusion

Summarize the key outcomes and potential future work.

References

[1] Zhang Jinrui. desinetask. `https://github.com/unjerry/autoCluster/blob/master/deSineTask.bat`. Accessed: 2024-10-21.

[2] Zhang Jinrui. Envreqs. `https://github.com/unjerry/autoCluster/blob/master/reqs.bat`. Accessed: 2024-10-21.

[3] Zhang Jinrui. Linear. `https://github.com/unjerry/autoCluster/blob/cbf0039/deLinearTasksSine.bat`. Accessed: 2024-10-21.

[4] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.