

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 [5, 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 [5, 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 $(\forall t)(F(\frac{d^2u}{dt^2}, \frac{du}{dt}, u) = 0)$.

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 $f, T_i, U_i, U_i^t, U_i^{tt}$

- 1: Initialize f randomly
 - 2: **repeat**
 - 3: $F_i \leftarrow f(U_i, U_i^t, U_i^{tt})$
 - 4: $RAND_i \leftarrow$ randomly sample in \mathbb{R}^3
 - 5: $R_i \leftarrow f(RAND_i)$
 - 6: $L \leftarrow meanSquareError(F_i, 0) + 0.1 * meanSquareError(R_i, 1)$
 - 7: backPropagation against L to optimize f
 - 8: **until** L meets requirement
 - 9: **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 $[U \ U^t \ U^{tt}]$ 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 $[U \ U^t \ U^{tt}]$ in the first

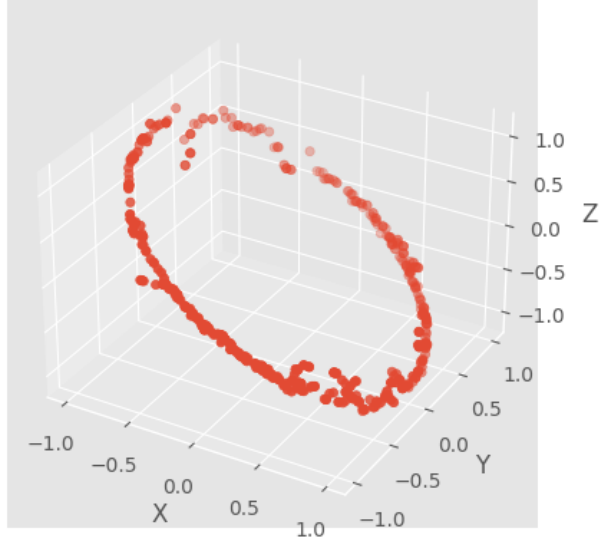


Figure 1: vector $[U \ U^t \ U^{tt}]$ sit on 1-dimensional manifold

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, deLinearTasks-Sine]

Algorithm 2 normal vector v calculator

Require: Input parameters U_i, U_i^t, U_i^{tt}
1: stack (U_i, U_i^t, U_i^{tt}) to latent vectors LAT_{ij}
2: perform $PCA(LAT_{ij})$
3: $v \leftarrow$ the eigen vector of the smallest singular value
4: **return** v

2.3 approach augmentation

This method is a autoencoder we can also check the difference between the input data and the data generate by the PINN in the differential relationship constrain. This can view as a parameterize method.

If the Manifold Hypothesis hold, a high dimensional dataset x_δ is a lower dimensional manifold ρ_d embedding in the high dimension space. Denote the dimension of the original data Δ , and denote the dimension of the latent variable D . To parameterize the data manifold, we need a decoder $\mathbb{R}^D \xrightarrow{f^{-1}} \mathbb{R}^\Delta$, $\rho_d \mapsto y_\delta$. The jacobian of f^{-1} denote as $J(f^{-1}) = J_d^\delta = \frac{\partial y_\delta}{\partial \rho_d}$. Denote the second order jacobian as $J^2(f^{-1}) = J_{d_1 d_2}^\delta = \frac{\partial^2 y_\delta}{\partial \rho_{d_1} \partial \rho_{d_2}}$. For the sake of symbolic coherence, denote $J^0(f^{-1}) = J^\delta$. To find a linear differential equation means to find equation in the form of $A_{d_1 d_2} J_{d_1 d_2}^\delta + A_d J_d^\delta + A J^\delta = 0$. The generate equation of order N could be written as $\sum_{j=0}^N A_{d_1 d_2 \dots d_j} J_{d_1 d_2 \dots d_j}^\delta = 0$. As we only want to find the direction of the hyper plane, the normal vector concatenate $(\forall j \leq N) A_{d_1 d_2 \dots d_j}$ denote as V , need to be normalize. To check this idea correctly, the V is a vector of dimension $\sum_{j=0}^N D^j$, so this algorithm may be computaional heavy to some higher order differential relationship encode, or to some high latent dimension encode task.

For the other side of the autoencoder, we donote the encoder as $\mathbb{R}^\Delta \xrightarrow{f} \mathbb{R}^D$, $x_\delta \mapsto \rho_d$. Normal autoencoder requires minimizing the mean square error between x_δ, y_δ . In the differential Informed method, the $A_{d_1 d_2} J_{d_1 d_2}^\delta + A_d J_d^\delta + A J^\delta$ term also need to be optimize to zero Algorithm3.

The experiment result can be obtained by Github code [4, deLinearAugSine]

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.

Algorithm 3 normal vector v calculator

Require: Input parameters $f, f^{-1}, x_\delta, (\forall j \leq N) A_{d_1 d_2 \dots d_j}$

```
1: Initialize  $f, f^{-1}$  randomly
2: repeat
3:    $\rho_d \leftarrow f(x_\delta)$ 
4:    $y_\delta \leftarrow f^{-1}(\rho_d)$ 
5:    $L \leftarrow \text{meanSquareError}(x_\delta, y_\delta)$ 
6:   backPropagation against  $L$  to optimize  $f, f^{-1}$ 
7: until  $L$  meets requirement
8: return  $f$ 
9: repeat
10:   $\rho_d \leftarrow f(x_\delta)$ 
11:   $y_\delta \leftarrow f^{-1}(\rho_d)$ 
12:   $(\forall j \leq N) A_{d_1 d_2 \dots d_j} \leftarrow \text{calculate the jacobian of each order}$ 
13:   $L \leftarrow \text{mse}(x_\delta, y_\delta) + \text{mse}(\sum_{j=0}^N A_{d_1 d_2 \dots d_j} J_{d_1 d_2 \dots d_j}^\delta, 0)$ 
14:  backPropagation against  $L$  to optimize  $f, f^{-1}, (\forall j \leq N) A_{d_1 d_2 \dots d_j}$ 
15: until  $L$  meets requirement
16: return  $f$ 
```

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.

3.3 approach augmentation

Train the model on a 2D circle and try to get a result of $x^\delta = y^\delta(\rho) = [\cos(\rho), \sin(\rho)]$, and the corresponding differential equation is $\frac{\partial^2 y_\delta}{\partial \rho^2} + y_\delta = 0$ Result of the latent space structure shows in Figure6 and Figure8, with the numerical result $0.7360 \frac{\partial^2 y_\delta}{\partial \rho^2} - 0.0328 \frac{\partial y_\delta}{\partial \rho} + 0.6761 y_\delta = 0$.

4 Conclusion

Summarize the key outcomes and potential future work.

To really get the data structure manifold in the high dimensional space is a hard task. In this article, two method was developed to get a dimension reduce algorithm with more ability to explanation.

The naive one is just compute the PCA of neighborhood of each point in the dataset, to find the latent differential structures as shown in Figure1, this can also be treated as the global check if the dataset meet the Manifold Hypothesis. With sufficient amount of data, the first method could learn basically all the nonlinear differential relationship in the dataset, and leverage the ability of PINN [5, PINN] to generate new data agree with the differential relationships.

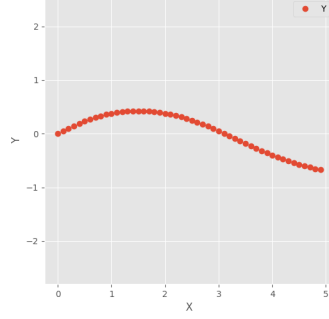


Figure 2: $0.5 * \sin(x)$

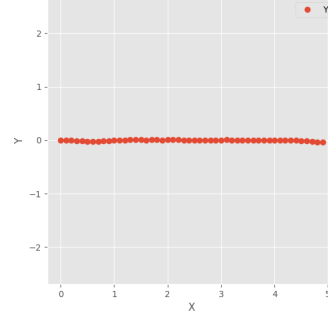


Figure 3: f errors

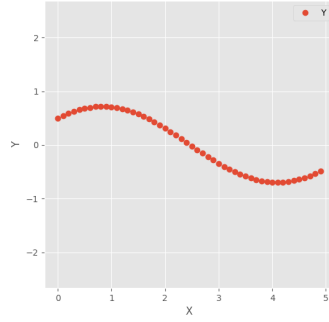


Figure 4: $\frac{\sqrt{2}}{2} * \sin(x + \frac{\pi}{4})$

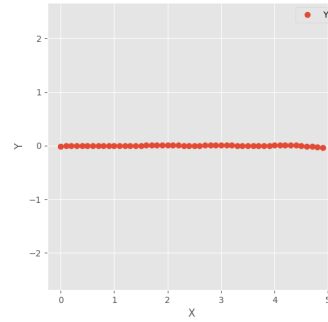


Figure 5: f errors

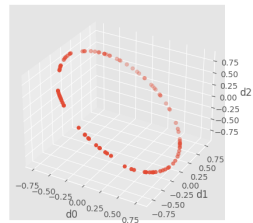


Figure 6: first component latent space of $y^\delta(\rho)$

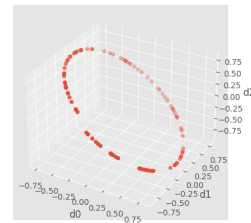


Figure 7: second component latent space of $y^\delta(\rho)$

The second method make a Linearity assumption, to significantly reduce the requirement of the amount of data. The augmentation of the Linearity assumption method, could be seen as a autoencoder who wants to find a parameterize

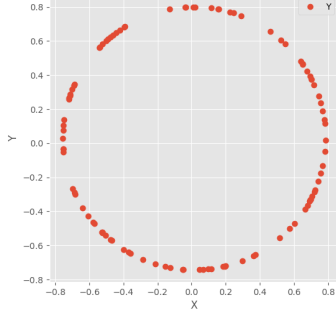


Figure 8: original data x_δ

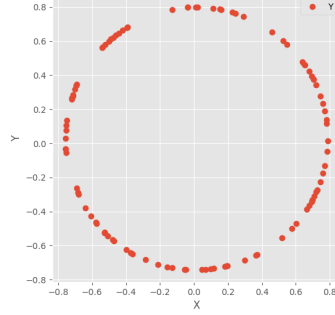


Figure 9: regenerate data y_δ

agree some Linear differential equations. With this method, the machine refound the *sin* and *cos* function from the circle dataset.

For the future work this model need to be test on more complicated data set agree with more intricate differential relationships. The jacobian for high dimensional latent space with high order differential relationship would require geometric amount of computaional complexity $\sum_{j=0}^N D^j$ which may be augmented or avoid by more advanced innovation in the future to have more efficiency.

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] Zhang Jinrui. Linearaug. <https://github.com/unjerry/autoCluster/blob/dabc8c85/deLinearAugTasksCircle.bat>. Accessed: 2024-10-23.
- [5] 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.