2456-8686, 6(1), 2022:117-123
https://doi.org/10.26524/cm125

ORIGINAL RESEARCH Open Access

Pseudo-Graph Neural Networks On Ordinary Differential Equations

B.Vembu[1],and Dr.S.Loghambal[2]

Received: 08 February 2022 / Accepted: 05 March 2022 / Published online: 22 March 2022

Abstract

In this paper, we extend the idea of continuous-depth models to pseudo graphs and present pseudo graph ordinary differential equations (PGODE), which are inspired by the neural ordinary differential equation (NODE) for data in the Euclidean domain. All existing graph networks have discrete depth. A pseudo graph neural network (PGNN) is used to parameterize the derivative of hidden node states, and the output states are the solution to this ordinary differential equation (ODE). A memory-efficient framework with precise gradient estimates is then proposed for free-form ODEs. We also introduce the framework of continuous–depth pseudo graph neural networks (PGNNs) on ODE by blending discrete structures and differential equations.

Key Words and Phrases: Pseudo graph, graph differential equation, Laplacian method, Pseudo Graph Neural Network (PGNN), Ordinal Differential Equation (ODE).

2010 Mathematics Subject Classifications: 05C12, 05C15, 05C20, 34A35, 34K40

1 Introduction

Convolutional neural networks (CNNs) have excelled at a variety of tasks, including image classification and segmentation, video processing and machine translation [1,3]. CNNs, on the other hand, are confined to data that can be represented by a grid in the Euclidean domain, such as images (2D grid) and text (1D grid), which prevents them from being used in datasets with irregular structures. Objects are represented as nodes in a graph

[1] *Research Scholar (Reg.No:19221072092002), PG & Research Department of Mathematics, The Madurai DiraviyamThayumanavar Hindu College,Tirunelveli 627 010, Tamil Nadu, India*. Email: vembu.iniya@gmail.com

[2] *Assistant Professor , PG & Research Department of Mathematics,The Madurai DiraviyamThayumanavar Hindu College,Tirunelveli 627 010, Tamil Nadu, India, E-mail: loghambalmdt2017@gmail.com*

data structure, while relationships between objects are represented as edges. Social networks and knowledge graphs are all examples of graphs being used to model irregularly organised data [8]. In [7] proposed a new class of models called graph neural networks (GNN). Researchers adapt convolution techniques to graphs to collect local information, inspired by the success of CNNs. Convolution on a graph can be done in two ways: spectral methods and non-spectral approaches. Spectral approaches often compute the graph Laplacian first, then do spectral domain filtering. For faster processing, other methods seek to approximate the filters without computing the graph Laplacian in [4].

To our knowledge, all of the above-mentioned GNN models have a discrete layer structure. The GNN struggles to describe continuous diffusion processes in graphs due to its discrete nature. A neural network is viewed as an ordinary differential equation (ODE) whose derivative is parameterized by the network, and the output is the solution to this ODE, according to the recently suggested neural ordinary differential equation (NODE) in [2]. We present pseudo graph ordinary differential equations (PGODE), which model message transmission on a graph as an ODE, by extending NODE from the Euclidean domain to graphs.

In this work, we show that this is due to an inaccuracy in gradient estimation during NODE training, and we present a memory-efficient framework for accurate gradient estimation. Our framework for free-form ODEs is shown to generalise to diverse model configurations and achieve better accuracy for both NODE and PGODE. Our contribution can be summarized as follows: 1. We propose a framework for free-form NODEs to accurately estimate the gradient, which is fundamental to deep-learning models. 2. Our framework is memory-efficient for free-form ODEs. 3. We generalize ODE to pseudo graph and propose PGNN on ODE.

2 Preliminaries

2.1 Differential Equations and Neural Networks

It has been proposed that neural networks be seen as differential equations. Based on an examination of the ODE, the neural ordinary differential equation (NODE) was proposed by Chen et al. in [2], which interprets the neural network as a continuous ODE. The adjoint technique has been utilised in optimal control and geophysical issues for a long time, and it has recently been applied to ODEs in [2]. To improve the expressive capability of NODEs which is developed into augmented neural ODEs. However, none of the methods above, to our knowledge, address the issue of erroneous gradient estimates; empirical results of NODE are much lower than state-of-the-art discrete-layer models.

2.2 Graph Neural Networks

Spectral and non-spectral GNNs are the two types of GNNs. Because spectral GNNs filter in the Fourier domain of a graph, they require information about the entire graph to calculate the graph Laplacian. Non-spectral GNNs, on the other hand, only evaluate message aggregation around neighbour nodes, making them confined and requiring less processing. First, we'll go through a few spectrum approaches in detail. Bruna et al. [9] were the first to introduce graph convolution in the Fourier domain based on the graph Laplacian, but due to non-localized filters, the computing cost is high. Defferrard et al. in [4] applied Chebyshev expansion to approximate the filters without having to compute the graph Laplacian and its eigenvectors, which resulted in a considerable speedup.

Fast localised spectral filtering on graphs was proposed in [4] and applied Chebyshev expansion to approximate the filters without having to compute the graph Laplacian and its eigenvectors, which resulted in a considerable speedup. In semi-supervised node classification tasks using a localised first-order approximation of graph convolution on graph data and achieved improved results. Fast localised spectral filtering on graphs was proposed by Defferrard et al. in [4]. Distinct weights are learned for different neighbours of a node in graph attention networks. The structure of the graph isomorphism network (GIN) is similar to that of the Weisfeiler-Lehman graph isomorphism test.

3. Accurate Gradient Estimation for Training of Node

3.1. Discrete Models to Continuous Models

First, we investigate the models for discrete layers with residual connection which can be expressed as follows:

$$a_{k+1} = a_k + f_k(a_k) \tag{1}$$

where a_k is the states in the kth layer; $f_k(.)$ is any function differentiated whose output is of the same form as the input.

If we add further layers with shared weights and allow steps in Eq. 1, the difference equation is converted into a neural ordinary differential equation (NODE):

$$\frac{dr(T)}{dT} = f(r(T), T) \tag{2}$$

We express hidden states with $r(T)$ in the continuous case and a_k in the discrete case. The derivative parameterized by a network is denoted as $f(.)$. The shape of f differs significantly between Eqs. 1 and 2: in the discrete case, individual layers (different k values) have their own function f_k, whereas in the continuous case, f is shared over all time T.

The forward pass of a discrete layer model can be stated as follows:

$$\begin{aligned} a_0 &= input, \\ a_1 &= a_0 + f_0(a_0), \\ a_2 &= a_1 + f_1(a_1), \ldots\ldots, \\ a_K &= a_{K-1} + f_{K-1}(a_{K-1}) \end{aligned} \tag{3}$$

where K is the total number of layers. Then an output layer (for example, fully-connected layer for classification) is then applied to a_K.

The forward pass of a NODE is:

$$r(t) = r(0) + \int_{T=0}^{t} \frac{dr(T)}{dT} dT = input + \int_{T=0}^{t} f(r(T), T) \tag{4}$$

where $r(0)$ is the input and t is the integration time, which in the discrete case corresponds to the number of layers K. The solution to the NODE is modelled as the transformation of states r. The output layer is then applied to $r(t)$.

4. On Pseudo Graph Networks, A Neural Ordinary Differential Equation

We start with pseudo graph neural networks with discrete layers, then move on to pseudo graph ordinary differential equations in the continuous case (PGODE).

4.1 Message Passing in PGNN

A pseudo graph is made up of nodes (shown by circles) and edges (represented by lines) (solid lines). Each node is given its own colour to make it easier to see. A message forwarding mechanism can be used to represent.

$$message_{(q,p)} = \phi^k(a_{k-1}^p, a_{k-1}^q, e_{(p,q)})$$

$$aggregation_p = \zeta_{q \in N(p)}(message_{(q,p)})$$

$$a_k^p = \gamma^k\left(a_{k-1}^p, aggregation_p\right) \quad (5)$$

where a_k^p is the states of the pth node in the network at the kth layer, and $e_{(p,q)}$ denotes the edge between nodes p and q. The set of neighbour nodes for node p is represented by N(p). represents a permutation invariant, differentiable operation like mean, max, or sum. Differentiable functions γ^k and ϕ^k are parameterized by neural networks.

A PGNN can be thought of as a three-stage model for a specific node p:

(1) Message passing, in which neighbour nodes q N(p) convey information to node p, denoted by message (q,p). The message is created by a neural network, which is parameterized by function $\phi(.)$.

(2) Message aggregation is defined as $aggregation_p$, where a node p collects all messages from its neighbours N(p). Because pseudo graphs are permutation invariant, the aggregation function is often permutation invariant operations like mean and sum.

(3) Notify, in which a node's states are updated based on its initial states a_{k-1}^p, and aggregate of messages $aggregation_p$, represented by γ(.).

4.2. Continuous-Time Models On Pseudo Graphs

By substituting f_p with the stated message passing process, which we term pseudo graph ordinary differential equation, we can convert a discrete-time PGNN to a continuous-time PGNN (PGODE). PGODE can capture extremely non-linear functions due to its nature as an ODE, and so has the potential to surpass its discrete-layer competitors. We show that PGODE's asymptotic stability is linked to the phenomenon of over-smoothing.

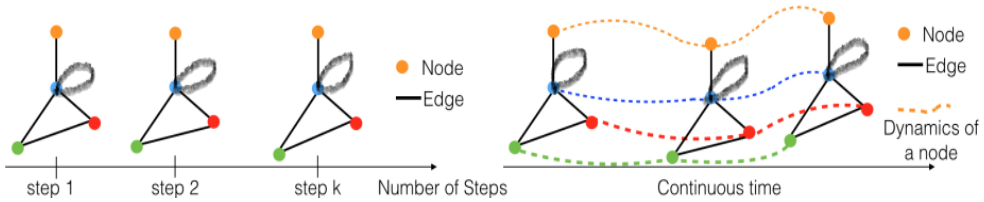
It is shown that pseudo graph convolution is a special case of Laplacian smoothing, which can be written as $B_p = (I_p - \gamma\bar{d}^{-1/2}\bar{l}\bar{d}^{-1/2})A_p$, where A_p *and* B_p are the input and output of a graph-conv layer, $X = X + I$ is the adjacency matrix, and d is the corresponding degree matrix of A, and is a positive scaling constant.

Also because symmetrically normalised Laplacian's eigenvalues are all real and non-negative, the above ODE's eigenvalues are also real and non-positive. Assume that the normalised Laplacian's eigenvalues are all non-zero. The ODE has only negative eigenvalues in this situation; hence the ODE is asymptotically stable. As a result, once t is large enough,

2456-8686, 6(1), 2022:117-123
https://doi.org/10.26524/cm125

all trajectories are close enough. This shows that if the integration time T is long enough, all nodes (from different classes) will have highly identical features, lowering the classification accuracy.

Figure 1:



(a) Discrete time model on pseudo graph (b)Continuous time model on pseudo graph

Figure 1: On a pseudo graph, discrete-time and continuous-time models. Nodes are represented by circles, with each node having a different colour. Solid lines are used to represent edges. The hidden states of nodes in discrete-time models (a) are updated in discrete increments. The hidden states of each node evolve continuously with time in continuous-time models (b). Node dynamics are depicted as dashed lines of the same colour as the related nodes.

5. Experiments

5.1. General Bijective Blocks

We show that the bijective blocks defined by

$$\begin{cases} y_2 = \psi(x_2, F_p(x_1)) \\ y_1 = \psi(x_1, G_p(y_2)) \end{cases}$$

are easily generalizable.

5.2. Theorem

For bijective blocks whose forward and reverse mappings are defined as,

$$\text{Forward } (m_1, n_2) = \begin{cases} n_2 = \psi(m_2, F_p(m_1)) \\ n_1 = \psi(m_1, G_p(n_2)) \end{cases}$$

$$\text{Reverse } (n_1, n_2) = \begin{cases} m_1 = \psi^{-1}(n_1, F_p(n_2)) \\ m_2 = \psi^{-1}(n_2, G_p(m_1)) \end{cases}$$

If $\psi(\alpha, \beta)$ is a bijective function with respect to α when β is given, then the block is a bijective mapping.

Proof. Proving that the forward mapping is bijective is the same as proving that it is both injective and surjective.

Injective: Here, we assume that, if $Forward\ (m_1, m_2) = Forward\ (m_3, n_4), then\ m_1 = m_3, m_2 = m_4$

The assumption is equal to $Forward\ (m_1, m_2) = Forward\ (m_3, m_4) \Leftrightarrow \begin{cases} n_2 = \psi(n_2, F_p(m_1)) = \psi(m_4, F_p(m_3)) \\ \quad = \psi(x_1, G_p(n_2)) = \psi(m_3, G_p(n_2)) \end{cases}$ (6)

We have $m_1 = m_3$ from equation $\frac{da(T)}{dT} = -a(T)^t \frac{\partial f(z(T),T,\theta}{z(T)}, \frac{d(L)}{d\theta} = -\int_T^0 a(T)^t \frac{\partial f(z(T),T,\theta}{\partial(\theta)} dt$ because $\psi(\alpha, \beta)$ is bijective with respect to when is supplied.
Using the bijective property of and condition on $m_1 = m_3$ and equation $(T) = \frac{\partial L}{\partial z(T)}$, applying ψ we get $m_2 = m_4$.
As a result, the mapping is injective in nature.

Surjective: We have to prove that, $\forall\ [n_1, n_2], \exists [m_1, m_2]$ s.t. $Forward\ (m_1, m_2) = [n_1, n_2]$

Now, we construct $m_1 = \psi^{-1}(n_1, G_p(n_2))\ , m_2 = \psi^{-1}(n_2, F_p(n_1))$. (7)

Then, given the bijective quality of, apply forward and reverse stated in the statement to the forward function.

$$r_2 = \psi(m_2, F_p(m_1)) = \psi(\ \psi^{-1}\left(n_2, F_p(m_1)\right) = F_p(m_1) = n_2$$
$$r_1 = \psi(m_1, G_p(n_2)) = \psi(\ \psi^{-1}\left(n_1, G_p(n_2)\right) = G_p(n_2) = n_1 \quad (8)$$

Hence, we construct m_1, m_2 s.t. $Forward\ (m_1, m_2) = [n_1, n_2]$. As a result, the mapping is purely subjective. As a result, it is bijective.

6. Conclusion

This article proposes PGNN on ODE, an application that enables us to model continuous diffusion on pseudo graphs, is created. We propose an experience direct back-propagation method for computing the gradient for universal free-form NODEs, and we show that it outperforms other methods on image classification and pseudo graph data. A relationship between PGNN over-smoothing and ODE asymptotic stability was also discovered.

References

[1] F. Scarselli, M. Gori, A. Chung Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. IEEE Transactions on Neural Networks, 20(1):61–80, 2008.

[2] K. He, X. Zhang, S. Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.

[3] L. Deng, D. Yu, et al. Deep learning: methods and applications. Foundations and Trends R in Signal Processing, 7(3–4):197–387, 2014.

[4] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In Advances in neural information processing systems, pp. 3844–3852, 2016.

[5] M. Fey and J.E. Lenssen. Fast graph representation learning with pytorch geometric, 2019.
[6] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data, 2015.
[7] P. Yanardag and S.V.N. Vishwanathan. Deep graph kernels. In ACM KDD, 2015.
[8] T.Q. Chen, Y. Rubanova, J. Bettencourt, and D.K. Duvenaud. Neural ordinary differential equations. In Advances in neural information processing systems, pp. 6571–6583, 2018.
[9] T. Hamaguchi, H. Oiwa, M. Shimbo, and Y. Matsumoto. Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach, 2017.