Poli et al. 2019: Graph Neural Ordinary Differential Equations

Minqi Pan

April 7, 2020

Graph Neural Ordinary Differential Equations

- AAAI 2020, "The 1st International Workshop on Deep Learning on Graphs: Methodologies and Applications", Feb 8th, 2020
- Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, Jinkyoo Park
- Korea Advanced Institute of Science and Technology, University of Tokyo

Outline

1. Background
   - Notation, GNN, Neural ODE and a Motivating Example
2. Graph Neural Ordinary Differential Equations
   - Static Models
   - Spatio-Temporal Continuous Graph Architectures
3. Experiments
   - Transductive Node Classification
   - Forecasting

Outline

1. Background
   - Notation, GNN, Neural ODE and a Motivating Example
2. Graph Neural Ordinary Differential Equations
   - Static Models
   - Spatio-Temporal Continuous Graph Architectures
3. Experiments
   - Transductive Node Classification
   - Forecasting

Notation

- $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- $|\mathcal{V}| = n$
- Adjacency matrix $A \in \mathbb{R}^{n \times n}$
- Feature vector $x_v(t) \in \mathbb{R}^d \quad \forall v \in \mathcal{V}$
- Feature matrix $X(t) \in \mathbb{R}^{n \times d}$
- $x_v(t), X(t)$ exhibits temporal dependencies

Neural ODE

Since Lu et al. 2018 (ICML 2018) and Chen et al. 2018 (NIPS 2018):

$$h_{s+1} = h_s + f(h_s, \theta), \quad s \in \mathbb{N}$$
$$\Downarrow$$
$$\frac{dh_s}{ds} = f(s, h_s, \theta), \quad s \in \mathcal{S} \subset \mathbb{R}$$

GNN+ODE

- Sanchez-Gonzalez et al. 2019: "Hamiltonian Graph Networks with ODE Integrators", combining graph networks with a differentiable ordinary differential equation integrator as a mechanism for predicting future states, and a Hamiltonian (the Hamiltonian in a physical/dynamical context) as an internal representation.
- Deng et al. 2019: "Continuous Graph Flow", a continuous normalizing flow model for graph generation

Static GNN

- Main variants:
  1. GCN (Kipf et al. 2016)
  2. DGC (Atwood et al. 2016)
  3. GAT (Veličković et al. 2017)
- Recurrent:
  1. GCRNN (Cui et al. 2018)
  2. GCGRU (Zhao et al. 2018)

A Motivating Example

- Multi–agent systems permeate science in a variety of fields
- Classical dynamical network theory since 2000s: nonlinear dynamical systems + graphs
- Often, closed–form analytic formulations are not available and forecasting or decision making tasks have to rely on noisy, irregularly sampled observations
- The primary purpose of "Graph Neural Ordinary Differential Equations" is to offer a data–driven approach to the modeling of dynamical networks, particularly when the governing equations are highly nonlinear and therefore challenging to approach with classical or analytical methods

Outline

1. Background
   - Notation, GNN, Neural ODE and a Motivating Example
2. Graph Neural Ordinary Differential Equations
   - Static Models
   - Spatio-Temporal Continuous Graph Architectures
3. Experiments
   - Transductive Node Classification
   - Forecasting

Inter–layer Dynamics of a GNN Node Feature Matrix

$$\begin{cases} H_{s+1} = H_s + F(s, H_s, \Theta_s) \\ H_0 = X \end{cases}, \quad s \in \mathbb{N}$$

- F: a matrix-valued nonlinear function conditioned on graph $\mathcal{G}$
- Θ_s: the tensor of trainable parameters of the s-th layer
- The explicit dependence on s of the dynamics is justified in DGC (Atwood et al. 2016)

Graph Neural Ordinary Differential Equation (GDE)

$$\begin{cases} \dot{H}_s = F(s, H_s, \Theta) \\ H_0 = X \end{cases}, \quad s \in S \subset \mathbb{R}$$

- A Cauchy problem
- $F : \mathcal{S} \times \mathbb{R}^{n \times d} \times \mathbb{R}^p \to \mathbb{R}^{n \times d}$ is a depth-varying vector field defined on graph $\mathcal{G}$

Well-posedness

- Let $\mathcal{S} \equiv [0, 1]$
- Under Lipschitz continuity of F w.r.t. H_s, and uniform continuity w.r.t. s
- The ODE admits a unique solution H_s defined in the whole $\mathcal{S}$
- There is a mapping Ψ from $\mathbb{R}^{n \times d}$ to the space of absolutely continuous functions $\mathcal{S} \to \mathbb{R}^{n \times d}$ such that $H \equiv \Psi(X)$ satisfies the ODE
- The output of the GDE:

$$\Psi(X) = X + \int_{\mathcal{S}} F(\tau, H_\tau, \Theta) d\tau$$

Integration Domain

- We restrict the integration interval to $\mathcal{S} \equiv [0, 1]$
- Any other integration time can be considered a rescaled version of $\mathcal{S}$
- In the forecasting with irregular timestamps application, where $\mathcal{S}$ acquires a specific meaning, the integration domain can be appropriately tuned to evolve GDE dynamics between arrival times without assumptions on underlying vector field (Rubanova et al. 2019)

GDE Training

- GDE can be trained with a variety of methods
  1. Standard backpropagation through the computational graph
  2. Adjoint methods for $O(1)$ memory efficiency
  3. Backpropagation through a relaxed spectral elements discretization (Quaglino et al. 2019)
- Numerical instability in the form of accumulating errors on the adjoint ODE during the backward pass of NODEs has been observed (Gholami et al. 2019)
  - A proposed solution is a hybrid checkpointing-adjoint scheme
  - the adjoint trajectory is reset at predetermined points in order to control the error dynamics

Incorporating Governing Differential Equations Priors

- GDEs belong to the toolbox of scientific deep learning along with Neural ODEs and other continuous depth models
- Scientific deep learning is concerned with merging prior, incomplete knowledge about governing equations with data-driven predictions
- GDEs can be extended to settings involving dynamical networks evolving according to different classes of differential equations

Stochastic Differential Equations

$$\begin{cases} dH_s = F(s, H_s)ds + G(s, H_s)dW_s \\ H_0 = X \end{cases}, \quad s \in \mathcal{S}$$

- F, G: GDEs that can be replaced by analytic terms when available
- W: a standard multidimensional Wiener process
- This extension enables a practical method to link dynamical network theory and deep learning with the objective of obtaining sample efficient, interpretable models

GCN: Graph Convolution Networks

$$H_{s+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H_s W_s)$$
$$\Downarrow$$
$$H_{s+1} = H_s + \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H_s W_s)$$
$$\Downarrow$$
$$\frac{dH}{ds} = F_{\mathsf{GCN}}(H, \Theta) \equiv \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H_s W_s)$$

- A skip connection is added

DGC: Diffusion Graph Convolution

$$H_{s+1} = H_s + \sigma(P^s X W_s)$$
$$\Downarrow$$
$$\frac{dH}{ds} = F_{\text{DGC}}(s, X, \Theta) \equiv \sigma(P^s X \Theta)$$

- $P \equiv D^{-1}A$: a probability transition matrix in $\mathbb{R}^{n \times n}$

Even Deeper

- While the definition of GDE models is given with F made up by a single layer
- In practice multi-layer architectures can also be used without any loss of generality
- In these models, the vector field defined by F is computed by considering wider neighborhoods of each node
- The paper's appendix also derives GDE counterparts of message passing neural networks and graph attention networks

Outline

1. Background
   - Notation, GNN, Neural ODE and a Motivating Example
2. Graph Neural Ordinary Differential Equations
   - Static Models
   - Spatio-Temporal Continuous Graph Architectures
3. Experiments
   - Transductive Node Classification
   - Forecasting

$s \equiv t$

- For settings involving a temporal component, the depth domain of GDEs coincides with the time domain and can be adapted depending on the requirements
- For example, given a time window Δt, the prediction performed by a GDE assumes the form

$$H_{t+\Delta t} = H_t + \int_t^{t+\Delta t} F(\tau, H_\tau, \Theta) d\tau$$

  regardless of the specific GDE architecture employed
- Here, GDEs represent a natural model class for autoregressive modeling of sequences of graphs $\{\mathcal{G}_t\}$ and directly fit into dynamical network theory

Hybrid Dynamical Systems

- Extending classical spatio-temporal architectures
- Hybrid Dynamical Systems: systems characterized by interacting continuous and discrete-time dynamics
- Let $(\mathcal{K}, >), (\mathcal{T}, >)$ be linearly ordered sets
- $\mathcal{K} \subset \mathbb{N}$
- $\mathcal{T} \equiv \{t_k\}_{k \in \mathcal{K}}$ is a set of time instances
- We suppose to be given a state-graph data stream which is a sequence in the form

$$\{(X_t, \mathcal{G}_t)\}_{t \in \mathcal{T}}$$

Hybrid Time Domain and Hybrid Arc

- Given $\{(X_t, \mathcal{G}_t)\}_{t \in \mathcal{T}}$
- Our aim is to build a continuous model predicting, at each $t_k \in \mathcal{T}$, the value of $X_{t_{k+1}}$
- Define a hybrid time domain:

$$\mathcal{I} \equiv \cup_{k \in \mathcal{K}}([t_k, t_{k+1}], k)$$

- Define a hybrid arc on $\mathcal{I}$ as a function Φ such that for each $k \in \mathcal{K}$, $t \mapsto \Phi(t, k)$ is absolutely continuous in $\{t : (t, j) \in \text{dom}\Phi\}$.

The Core Idea

- The core idea is to have a GDE smoothly steering the latent node features between two time instants
- And then apply some discrete operator, resulting in a "jump" of H
- H is then processed by an output layer
- Therefore solutions of the proposed continuous spatio-temporal model are hybrid arcs

Autoregressive GDEs (1)

$$\begin{cases} \dot{H}_s = F(H_s, \Theta), & s \in [t_k, t_{k+1}] \\ H_s^+ = G(H_s, X_{t_k}), & s = t_{k+1}, k \in \mathcal{K} \\ Y_{t_{k+1}} = K(H_s) \end{cases}$$

- F, G, K: GNN-like operators or general neural network layers
- H^+: the value of H after the discrete transition

Autoregressive GDEs (2)

$$\begin{cases} \dot{H}_s = F(H_s, \Theta), & s \in [t_k, t_{k+1}] \\ H_s^+ = G(H_s, X_{t_k}), & s = t_{k+1}, k \in \mathcal{K} \\ Y_{t_{k+1}} = K(H_s) \end{cases}$$

- Compared to standard recurrent models which are only equipped with discrete jumps, this system incorporates a continuous flow of latent node features H between jumps
- This feature of autoregressive GDEs allows them to track dynamical systems from irregular observations
- Different combinations of F, G, K can yield continuous variants of most common spatio-temporal GNN models
- F, G, K can themselves have multi-layer structure

E.g. Graph Differential Convolutional GRU

$$\begin{cases} \dot{H}_s = F_{\mathsf{GCN}}(H_s), & s \in [t_k, t_{k+1}] \\ H_s^+ = \mathsf{GCGRU}(H_s, X_{t_k}), & s = t_{k+1}, k \in \mathcal{K} \\ Y_{t_{k+1}} = \sigma(W H_s + b) \end{cases}$$

- W: a learnable weight matrix

Outline

1. Background
   - Notation, GNN, Neural ODE and a Motivating Example
2. Graph Neural Ordinary Differential Equations
   - Static Models
   - Spatio-Temporal Continuous Graph Architectures
3. Experiments
   - Transductive Node Classification
   - Forecasting

Experimental Setup

- Static graphs (Cora, PubMed, CiteSeer)
- Semi-supervised
- Transductive
- Node classification
- Goal: show the usefulness of GDEs as general GNNs variants even when the data is NOT generated by continuous dynamical systems

Discussion

- Mean and standard deviation across 100 training runs are reported
- GCDE–rk4 outperforms GCNs across all datasets
- Accuracy and training stability improved
- GCDEs do not require more parameters than their discrete counterparts
- NEW "depth": the number of function evaluations (NFE) of the ODE function
- 108-depth GCDE-dpr5 is slightly worse compared to 4-depth GCDE–rk4, since deeper models are penalized on these datasets by a lack of sufficient regularization
- These are arXiv v1 (Nov 2019) results. v2 adds a tuned GCN*: GCDE–rk4 83.8/72.5/79.5 vs. GCN* 82.8/71.2/79.5 (Cora/Citeseer/PubMed); GCDE–dpr5 (158 NFE) falls below GCN*

Outline

1. Background
   - Notation, GNN, Neural ODE and a Motivating Example
2. Graph Neural Ordinary Differential Equations
   - Static Models
   - Spatio-Temporal Continuous Graph Architectures
3. Experiments
   - Transductive Node Classification
   - Forecasting

Experimental Setup

- Dataset: PeMS7(M), a subsampled version of PeMS obtained via selection of 228 sensor stations and aggregation of their historical speed data into regular 5 minute frequency time series
- With missing data and irregular timestamps: undersample the time series by performing independent Bernoulli trials on each data point with probability 0.7 of removal
- Comparison: in order to measure performance gains obtained by GDEs in settings with data generated by continuous time systems, we employ a GCDE–GRU as well as its discrete counterpart GCGRU (Zhao, Chen, and Cho 2018)

Discussion (1)

- The delta time scale $t_{k+1} - t_k$ of required predictions used to adjust the ODE integration domain of GCDE-GRU varies greatly during the task
- Non-constant differences between timestamps result in a challenging forecasting task for a single model since the average prediction horizon changes drastically over the course of training and testing
- For a fair comparison between models we include delta timestamps information as an additional node feature for GCGRUs and GRUs

Discussion (2)

- The main objective of these experiments is to measure the performance gain of GDEs when exploiting a correct assumption about the underlying data generating process
- Traffic systems are intrinsically dynamic and continuous and therefore a model able to track continuous underlying dynamics is expected to offer improved performance
- Since GCDE-GRUs and GCGRUs are designed to match exactly in structure and number of parameters we can measure this performance increase

Discussion (3)

- GDEs offer an average improvement of 3% in normalized RMSE and 7% in mean absolute percentage error
- A variety of other application areas with continuous dynamics and irregular datasets could similarly benefit from adopting GDEs as modeling tools: medicine, finance or distributed control systems, to name a few.