Updates and Progress

Yogesh Verma
Doctoral Candidate
Aalto University

Updates

- Paper read: 61/1280
  - Auto-regressive diffusion models
  - GRAND++
  - Reimanian Neural SDE
  - Message Passing Neural PDE Solvers
  - Temporal Graph Networks
  - Neural Sheaf Diffusion [Reading]
  - Critical points in Quantum Generative Models [Reading]

Updates

- Paper read: 61/1280
  - Auto-regressive diffusion models
  - GRAND++
  - Reimanian Neural SDE
  - Message Passing Neural PDE Solvers
  - Temporal Graph Networks
  - Neural Sheaf Diffusion [Reading]
  - Critical points in Quantum Generative Models [Reading]
- Reversible SDEs for graphs
- Ideas in Stack: **Next on List**
- TO DO: External Supervisor (CS Doctoral Support Program)

Next on the List:

Next on the List:

- Supervised QSAR Modlfow:
  - Include property prediction (eg. predict y=logp)
  - Let the property variable also be a state that evolves as $y(0) \rightarrow y(T)$ (Continual Learning?)

Next on the List:

- Supervised QSAR Modlfow:
  - Include property prediction (eg. predict y=logp)
  - Let the property variable also be a state that evolves as $y(0) \to y(T)$ (Continual Learning?)
- Modular Diffusion for Conformer Generation (Underway)

Next on the List:

- Supervised QSAR Modlfow:
  - Include property prediction (eg. predict y=logp)
  - Let the property variable also be a state that evolves as $y(0) \rightarrow y(T)$ (Continual Learning?)
- Modular Diffusion for Conformer Generation (Underway)
- CLoud Flow
  - Instead of graphs, let's evolve continuous space-time function surfaces u(x,y,z,t)
    - ★ Option 1: represent "u" with splines/kernels/rbfs/fouriers, evolve their parameters (weight coefficients in eigen basis)
    - ★ Option 2: evolve sampled particle cloud NF-style

Next on the List:

- Supervised QSAR Modlfow:
  - Include property prediction (eg. predict y=logp)
  - Let the property variable also be a state that evolves as $y(0) \rightarrow y(T)$ (Continual Learning?)
- Modular Diffusion for Conformer Generation (Underway)
- CLoud Flow
  - Instead of graphs, let's evolve continuous space-time function surfaces u(x,y,z,t)
    - Option 1: represent "u" with splines/kernels/rbfs/fouriers, evolve their parameters (weight coefficients in eigen basis)
    - Option 2: evolve sampled particle cloud NF-style
  - Apply ModFlow to 3D molecule clouds (evolving electron/mass blobs/shapes, no more "nodes") (modeling molecular dynamics??)
  - Apply ModFlow to climate data time series forecasting on the spherical globe (Possible exploration, *pie-in-the-sky*)

Reversible SPDE on Graphs for Conformer Generation

Before we go,

- Now, we are working in R^3 space $\rightarrow$ Coordinate embeddings (No argmax.....)

Before we go,

- Now, we are working in R^3 space $\rightarrow$ Coordinate embeddings (No argmax.....)
- Using SDEs (explained later) to generate conformers of molecules, use spherical polar coordinates to use 3D geometric information

Before we go,

- Now, we are working in R^3 space $\rightarrow$ Coordinate embeddings (No argmax.....)
- Using SDEs (explained later) to generate conformers of molecules, use spherical polar coordinates to use 3D geometric information
- Coordinate embeddings as node features ($\mathbf{x}$)...

SDE

An Ito SDE can be written as:

$$d\mathbf{X}_t = \mathbf{f}_t(\mathbf{X}_t)dt + \mathbf{g}_t(\mathbf{X}_t)d\mathbf{w} \tag{1}$$

where $\mathbf{f}_t$ is the drift coefficient, $\mathbf{g}_t$ is diffusion coefficient and $\mathbf{w}$ is standard weiner process.

SDE

An Ito SDE can be written as:

$$d\mathbf{X}_t = \mathbf{f}_t(\mathbf{X}_t)dt + \mathbf{g}_t(\mathbf{X}_t)d\mathbf{w} \tag{1}$$

where $\mathbf{f}_t$ is the drift coefficient, $\mathbf{g}_t$ is diffusion coefficient and $\mathbf{w}$ is standard weiner process.
The reverse-time SDE for above can be written as (Ref)

$$d\mathbf{X}_t = [\mathbf{f}_t(\mathbf{X}_t) - \mathbf{g}_t^2 \nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t)]d\tilde{t} + \mathbf{g}_t(\mathbf{X}_t)d\tilde{\mathbf{w}} \tag{2}$$

where $\tilde{w}$ is reverse-time standard wiener process and $d\tilde{t}$ is an infinitesimal negative time step.

SDE on Graphs

A graph $\mathbf{G}$ can represented by $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X})$ where $\mathbf{X}$ are the node $(\mathbf{V})$ features and $\mathbf{E}$ are the edges determining the connections.

SDE on Graphs

A graph $\mathbf{G}$ can represented by $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X})$ where $\mathbf{X}$ are the node $(\mathbf{V})$ features and $\mathbf{E}$ are the edges determining the connections.
The forward diffusion process can be represented with continuous time variable $t \in [0, T]$ where $X_0 \sim p_{data}$ and $X_T \sim p_T$

$$d\mathbf{X}_t = \mathbf{f}_t(\mathbf{X}_t)dt + \mathbf{g}_t(\mathbf{X}_t)d\mathbf{w} \tag{3}$$

SDE on Graphs

A graph $\mathbf{G}$ can represented by $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X})$ where $\mathbf{X}$ are the node $(\mathbf{V})$ features and $\mathbf{E}$ are the edges determining the connections.
The forward diffusion process can be represented with continuous time variable $t \in [0, T]$ where $X_0 \sim p_{data}$ and $X_T \sim p_T$

$$d\mathbf{X}_t = \mathbf{f}_t(\mathbf{X}_t)dt + \mathbf{g}_t(\mathbf{X}_t)d\mathbf{w} \tag{3}$$

Following the same analogy, the reverse process can be defined as

$$d\mathbf{X}_t = [\mathbf{f}_t(\mathbf{X}_t) - \mathbf{g}_t^2 \nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t)]d\tilde{t} + \mathbf{g}_t(\mathbf{X}_t)d\tilde{\mathbf{w}} \tag{4}$$

SDE on Graphs

- Assuming a 1-neighbourhood where local effects are strong, we can factorize or decompose $\mathbf{f}_t(\mathbf{X}_t)$ as contribution from local regions ($\mathbf{x}_t^v$ is the node features of v node at time t)

$$\mathbf{f}_t(\mathbf{X}_t) = \mathtt{Agg}_{\mathbf{v}\in V}(\mathbf{f}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v))) \tag{5}$$

SDE on Graphs

- Assuming a 1-neighbourhood where local effects are strong, we can factorize or decompose $\mathbf{f}_t(\mathbf{X}_t)$ as contribution from local regions ($\mathbf{x}_t^v$ is the node features of v node at time t)

$$\mathbf{f}_t(\mathbf{X}_t) = \mathtt{Agg}_{\mathbf{v}\in V}(\mathbf{f}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v))) \tag{5}$$

- By using chain rule of differentiation and factorization we can write

$$\nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t) = \frac{\partial \log p_t(\mathbf{X}_t)}{\partial \mathbf{X}_t} := \sum_{\mathbf{v}\in V} \sum_{v`\in v\cup\mathcal{N}(v)} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v} \tag{6}$$

SDE on Graphs

- One can decompose $\mathbf{g}_t(\mathbf{X}_t)$ similarly as,

$$\mathbf{g}_t^2 \nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t) = \sum_{\mathbf{v} \in V} \mathbf{g}_t^2 \sum_{v` \in v \cup \mathcal{N}(v)} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^{v}} \tag{7}$$

SDE on Graphs

- One can decompose $\mathbf{g}_t(\mathbf{X}_t)$ similarly as,

$$\mathbf{g}_t^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{X}_t) = \sum_{\mathbf{v} \in V} \mathbf{g}_t^2 \sum_{v` \in v \cup \mathcal{N}(v)} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v} \tag{7}$$

- Now one can use above equations in Eq.9 and decompose it for each $\mathbf{x} \in X$ as

$$d\mathbf{x}_t^v = [\mathbf{f}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v)) - \mathbf{g}_t^2 \sum_{v` \in v \cup \mathcal{N}(v)} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v}] d\tilde{t} + \mathbf{g}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v)) d\tilde{\mathbf{w}} \tag{8}$$

Connection with other variants

- One can establish connection with other GNN variants, such as:

Connection with other variants

- One can establish connection with other GNN variants, such as:
  - Temporal/Spatial Message Passing Neural Networks

Connection with other variants

- One can establish connection with other GNN variants, such as:
  - Temporal/Spatial Message Passing Neural Networks
  - Graph Convolutional Neural Networks

Connection with other variants

- One can establish connection with other GNN variants, such as:
  - Temporal/Spatial Message Passing Neural Networks
  - Graph Convolutional Neural Networks
  - Graph Attention Networks, etc..

How to do?

- Model the score when training with score matching
- Sampling with score-based MCMC like Langevin MCMC, HMC (Similar to multiple noise levels for greater accuracy in low density regions as well in Song et al. 2020)

Factorized score matching

- Since, there occurs a factorization in the probability due to local neighbourhood dependence. This also leads to factorized score matching as we now only need to approximate $\sum_{v‘\in v\cup\mathcal{N}(v)} \frac{\partial \log p_t(\mathbf{x}_t^{v'}|\mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^{v}}$ for node v

Factorized score matching

- Since, there occurs a factorization in the probability due to local neighbourhood dependence. This also leads to factorized score matching as we now only need to approximate $\sum_{v`\in v\cup\mathcal{N}(v)} \frac{\partial \log p_t(\mathbf{x}_t^{v'}|\mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v}$ for node v
- Plus points:
  - Less dimensionality of the score conditional $\rightarrow$ easy to accurately approximate in contrast to currently used techniques on images

Factorized score matching

- Since, there occurs a factorization in the probability due to local neighbourhood dependence. This also leads to factorized score matching as we now only need to approximate $\sum_{v`\in v\cup\mathcal{N}(v)} \frac{\partial \log p_t(\mathbf{x}_t^{v'}|\mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v}$ for node v
- Plus points:
  - Less dimensionality of the score conditional $\rightarrow$ easy to accurately approximate in contrast to currently used techniques on images
  - Langevin MCMC sampling giving structure constrained sampling (can also be extended to HMC sampling)

$$\mathbf{x}_{i+1}^v \leftarrow \mathbf{x}_i^v + \epsilon \sum_{v`\in v\cup\mathcal{N}(v)} \frac{\partial \log p_t(\mathbf{x}_t^{v'}|\mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v} + \sqrt{2\epsilon}\mathbf{z}_i \tag{9}$$

Modular Adjoints over SDE

- One can extend the modular joints from previous project to modular joints over SDE

Modular Adjoints over SDE

- One can extend the modular joints from previous project to modular joints over SDE
- Leads to Augmented Diffusion, Drift for each node as we decompose f and g

THANK YOU

FEEDBACK?