Updates and Progress

Yogesh Verma
Doctoral Candidate
Aalto University

Updates

- Paper read: 42/1280
  - GRAND: Graph Neural Diffusion
  - DATA-EFFICIENT GRAPH GRAMMAR LEARNING FOR MOLECULAR GENERATION
  - LEARNING ENERGY-BASED MODELS BY DIFFUSION RECOVERY LIKELIHOOD
  - Spherical message passing for 3D molecular graphs [Reading]
  - Learning to extend molecular scaffolds with structural motifs [Reading]

Updates

- Paper read: 42/1280
  - GRAND: Graph Neural Diffusion
  - DATA-EFFICIENT GRAPH GRAMMAR LEARNING FOR MOLECULAR GENERATION
  - LEARNING ENERGY-BASED MODELS BY DIFFUSION RECOVERY LIKELIHOOD
  - Spherical message passing for 3D molecular graphs [Reading]
  - Learning to extend molecular scaffolds with structural motifs [Reading]
- CNF for generating valid molecules, coding, debugging.....
- Reversible SDEs for graphs
- Ideas in Stack: Molecular surface by 3D Zerneike Descriptors

Proposed Method

- Aim: Learn realistic molecular distributions and generating valid molecules
- We represent atom configurations by modelling local neighborhoods with coupled PDE CNFs in graph domain, Attack validity by accurate local densities

Proposed Method

- Aim: Learn realistic molecular distributions and generating valid molecules
- We represent atom configurations by modelling local neighborhoods with coupled PDE CNFs in graph domain, Attack validity by accurate local densities
- Given a molecule with a graph representation (connectivity C) $\mathcal{G} = (V, E, X)$, one can define a probability distribution at each node over the vocabulary, conditioned over each node neighbours $\mathcal{N}(\mathbf{v})$.

$$P(G) = \prod_{\mathbf{v} \in V} p\left(\mathbf{x}_v \mid \mathbf{x}_{\mathcal{N}(\mathbf{v})}\right) \tag{1}$$

- Building on CNFs, we present flows on graphs specially applied in the molecular regime where we model the continuous time dynamics of random variables on graphs with respect to some conditionals over connectivity of the graph applied to graph structured data

Proposed Method

- Given a set of vertices $\mathbf{V}$ and its features $\mathbf{X}$ for a graph (molecule), the goal is to learn the joint distribution $P(G)$ given by Eq.1 .
- For continuous time dynamics of each $\mathbf{v} \in V$, by following Eq. 3,4 we formulate an ODE system as follows

$$\frac{\partial \mathbf{x}_v}{\partial t} = \mathrm{f}\left(\mathbf{x}_v, \mathbf{x}_{\mathcal{N}(\mathbf{v})}, t\right) \tag{2}$$

Proposed Method

- Given a set of vertices $\mathbf{V}$ and its features $\mathbf{X}$ for a graph (molecule), the goal is to learn the joint distribution $P(G)$ given by Eq.1 .
- For continuous time dynamics of each $\mathbf{v} \in V$, by following Eq. 3,4 we formulate an ODE system as follows

$$\frac{\partial \mathbf{x}_v}{\partial t} = \mathrm{f}\left(\mathbf{x}_v, \mathbf{x}_{\mathcal{N}(\mathbf{v})}, t\right) \tag{2}$$

- Then the change in log probability follows

$$\frac{\partial \log p\left(\mathbf{x}_v(t), t\right)}{\partial t} = -tr\left(\frac{\partial \mathrm{f}\left(\mathbf{x}_v, \mathbf{x}_{\mathcal{N}(\mathbf{v})}, t\right)}{\partial \mathbf{x}_v(t)}\right) \tag{3}$$

f and Optimization

- Options for f
  - NN

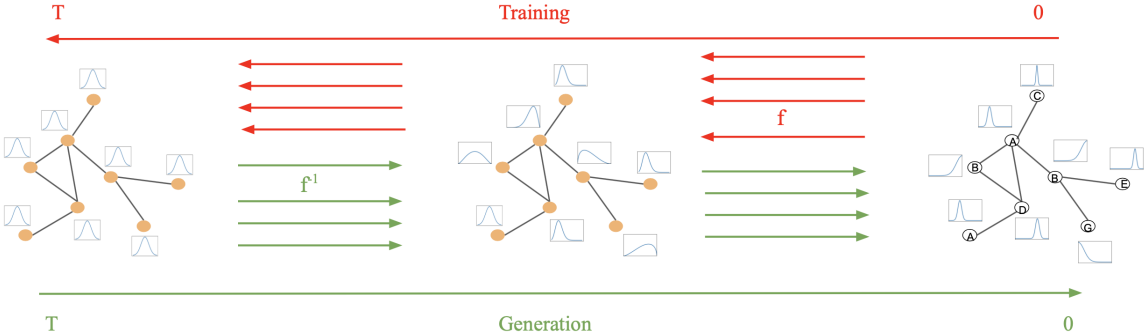
$$f = \text{NN}(\mathbf{x}_v, \mathbf{x}_{\mathcal{N}(\mathbf{v})}) \tag{4}$$

  - RBF+NN: Transform the neighborhood into a distribution ($\sum_{\mathbf{v}_j \in \mathcal{N}(\mathbf{v}_i)} \phi(\mathbf{x}_{v_i}, \mathbf{x}_{v_j})$), and then input the distribution into a NN

f and Optimization

- Options for f
  - NN
  $$f = \text{NN}(\mathbf{x}_v, \mathbf{x}_{\mathcal{N}(\mathbf{v})}) \tag{4}$$
  - RBF+NN: Transform the neighborhood into a distribution ($\sum_{\mathbf{v}_j \in \mathcal{N}(\mathbf{v}_i)} \phi(\mathbf{x}_{v_i}, \mathbf{x}_{v_j})$), and then input the distribution into a NN
- Fit the flow-based model to the samples by maximum likelihood estimation

Workflow



Comparison

Method	Continous time	Invertible	Scalable	Non-Iterative Sampling
JT-VAE	✗	✗	✗	✓
RVAE	✗	✗	✗	✓
GraphNVP	✗	✓	✓	✓
GraphAF	✗	✓	✓	✗
GraphDF	✗	✓	✓	✗
Ours	✓	✓	✓	✓

Expansions: Junction Tree

- A tree decomposition maps a graph $\mathcal{G}$ into a junction tree by contracting certain vertices into a single node.

Expansions: Junction Tree

- A tree decomposition maps a graph $\mathcal{G}$ into a junction tree by contracting certain vertices into a single node.
- For a given graph $\mathcal{G}$, a junction tree $\mathcal{T}_{\mathcal{G}} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ is a connected tree where $\mathcal{V} = (C_1, C_2, ..., C_n)$ and $\mathcal{E}$ are corresponding node and edge set.

Expansions: Junction Tree

- Consider only ring structures to be choice for clusters

Expansions: Junction Tree

- Consider only ring structures to be choice for clusters
- Analysis: Nearly $\sim$ 2000 unique rings for 100k ZINC250K data

Expansions: Junction Tree

- Consider only ring structures to be choice for clusters
- Analysis: Nearly $\sim$ 2000 unique rings for 100k ZINC250K data
- But, $\sim$ 20 has high frequency of appearing and most have only single instance $\rightarrow$ skewed distribution

Expansions: Junction Tree

- Consider only ring structures to be choice for clusters
- Analysis: Nearly $\sim$ 2000 unique rings for 100k ZINC250K data
- But, $\sim$ 20 has high frequency of appearing and most have only single instance $\rightarrow$ skewed distribution
- Consider only first 10 or 20 high frequency rings appearing and train the model

Expansions: Dependent sampling

- Using Conditional random fields as a special case of MRF, where we can model the conditional distribution in a factor graph, we can model the conditional neighbourhood of a node label (y) given the neighbours in final state as

$$p(\mathbf{y}_v|\mathbf{x}_{\mathcal{N}(\mathbf{v})}) = CRF(\lambda, f) \tag{5}$$

Expansions: Dependent sampling

- Using Conditional random fields as a special case of MRF, where we can model the conditional distribution in a factor graph, we can model the conditional neighbourhood of a node label (y) given the neighbours in final state as

$$p(\mathbf{y}_v|\mathbf{x}_{\mathcal{N}(\mathbf{v})}) = CRF(\lambda, f) \tag{5}$$

- Leads to a dependent sampling when we are given the final probability tensor
- CRF itself has to be trained by MLE, can be Incorporated within the whole model

Expansions: One-point posterior

- Aim: Replace Argmax
- A categorical graph likelihood can be written as:

$$p(G|\phi) = \prod_{v \in G} \mathrm{Cat}(v|\phi) \prod_{e \in G} \mathrm{Cat}(e|\phi), \tag{6}$$

Expansions: One-point posterior

- Aim: Replace Argmax
- A categorical graph likelihood can be written as:

$$p(G|\phi) = \prod_{v \in G} \mathrm{Cat}(v|\phi) \prod_{e \in G} \mathrm{Cat}(e|\phi), \tag{6}$$

- Normalizing flow model where the parameters ϕ are a r.v. with fixed initial distribution, propose a continuous-time flow to evolve the parameters with neural network parameters θ over time $t \in \mathbb{R}_+$,

$$\frac{d\phi_t}{dt} = \dot{\phi}_t = f(\phi_t; \theta), \qquad t \in [0, T]. \tag{7}$$

$$\log p_T(\phi_T) = \log p_0(\phi_0) - \int_0^T \mathrm{tr} \frac{\partial f(\phi_t)}{\partial \phi_t} dt. \tag{8}$$

Expansions: One-point posterior

- Marginal Likelihood can be given as ,where $\{G_n\} \sim p(G)$ is the set of observed graph samples

$$\mathbb{E}_{p(G)}\mathbb{E}_{p(\phi_T)}[p(G|\phi_T)] \approx \sum_{n=1}^{N} \int p(G_n|\phi_T) p_T(\phi_T) d\phi_T \tag{9}$$

$$= \sum_{n=1}^{N} \int p(G_n|\mathrm{solve}(\phi_0)) \exp\left(\log p_0(\phi_0) - \int_0^T \mathrm{tr} \frac{\partial f(\phi_t)}{\partial \phi_t}\right) d\phi_0 \tag{10}$$

- Contribution for one graph G_n from multiple parameter endpoints ϕ_T, and similarly one parameter ϕ_T contributes to likelihood of multiple graphs G_n.

Expansions: One-point posterior

- Computationally expensive, look at all graph-parameter (G, ϕ) pairs

Expansions: One-point posterior

- Computationally expensive, look at all graph-parameter (G, ϕ) pairs
- Sample all parameters ϕ compatible with one graph G by the inverse likelihood distribution

$$\phi \sim p(G_n|\phi) \tag{11}$$
$$\phi \sim p(\phi|G_n)p(G_n) \tag{12}$$

- Draw samples from conditional $\phi \sim p(\phi|G_n)$, and then MC average Eq. 11

Reversible SDE on Graphs

SDE

An Ito SDE can be written as:

$$d\mathbf{X}_t = \mathbf{f}_t(\mathbf{X}_t)dt + \mathbf{g}_t(\mathbf{X}_t)d\mathbf{w} \tag{13}$$

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:

$$d\mathbf{X}_t = \mathbf{f}_t(\mathbf{X}_t)dt + \mathbf{g}_t(\mathbf{X}_t)d\mathbf{w} \tag{13}$$

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)

$$d\mathbf{X}_t = [\mathbf{f}_t(\mathbf{X}_t) - \mathbf{g}_t^2 \nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t)]d\tilde{t} + \mathbf{g}_t(\mathbf{X}_t)d\tilde{\mathbf{w}} \tag{14}$$

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 as $\{\mathbf{G}_t = (\mathbf{X}_t)\}_{t=0}^{T}$ [assuming $\mathbf{E}$ remain same during time] with continuous time variable $t \in [0, T]$ where $X_0 \sim p_{data}$ and $X_T \sim p_T$

$$d\mathbf{X}_t = \mathbf{f}_t(\mathbf{X}_t)dt + \mathbf{g}_t(\mathbf{X}_t)d\mathbf{w} \tag{15}$$

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 as $\{\mathbf{G}_t = (\mathbf{X}_t)\}_{t=0}^{T}$ [assuming $\mathbf{E}$ remain same during time] with continuous time variable $t \in [0, T]$ where $X_0 \sim p_{data}$ and $X_T \sim p_T$

$$d\mathbf{X}_t = \mathbf{f}_t(\mathbf{X}_t)dt + \mathbf{g}_t(\mathbf{X}_t)d\mathbf{w} \tag{15}$$

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

$$d\mathbf{X}_t = [\mathbf{f}_t(\mathbf{X}_t) - \mathbf{g}_t^2 \nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t)]d\tilde{t} + \mathbf{g}_t(\mathbf{X}_t)d\tilde{\mathbf{w}} \tag{16}$$

SDE on Graphs

- The reverse process can be defined as

$$d\mathbf{X}_t = [\mathbf{f}_t(\mathbf{G}_t) - \mathbf{g}_t^2 \nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t)] d\tilde{t} + \mathbf{g}_t(\mathbf{X}_t) d\tilde{\mathbf{w}} \tag{17}$$

SDE on Graphs

- The reverse process can be defined as

$$d\mathbf{X}_t = [\mathbf{f}_t(\mathbf{G}_t) - \mathbf{g}_t^2 \nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t)] d\tilde{t} + \mathbf{g}_t(\mathbf{X}_t) d\tilde{\mathbf{w}} \tag{17}$$

- 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)

$$\mathbf{f}_t(\mathbf{X}_t) = \texttt{Agg}_{\mathbf{v} \in V}(\mathbf{f}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v))) \tag{18}$$

SDE on Graphs

- By using chain rule of differentiation we can write

$$\nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t) = \frac{\partial \log p_t(\mathbf{X}_t)}{\partial \mathbf{X}_t} = \sum_{\mathbf{v} \in V} \frac{\partial \log p_t(\mathbf{X}_t)}{\partial \mathbf{x}_t^v} \frac{\partial \mathbf{x}_t^v}{\partial \mathbf{X}_t} = \sum_{\mathbf{v} \in V} \nabla_{\mathbf{x}_t^v} \log p_t(\mathbf{X}_t) \cdot \frac{\partial \mathbf{x}_t^v}{\partial \mathbf{X}_t} \quad (19)$$

- $\frac{\partial \mathbf{x}_t^v}{\partial \mathbf{X}_t}$ similar to gradient of node w.r.t graph (maybe a connection to laplacian)

SDE on Graphs

- By using chain rule of differentiation we can write

$$\nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t) = \frac{\partial \log p_t(\mathbf{X}_t)}{\partial \mathbf{X}_t} = \sum_{\mathbf{v} \in V} \frac{\partial \log p_t(\mathbf{X}_t)}{\partial \mathbf{x}_t^v} \frac{\partial \mathbf{x}_t^v}{\partial \mathbf{X}_t} = \sum_{\mathbf{v} \in V} \nabla_{\mathbf{x}_t^v} \log p_t(\mathbf{X}_t) \cdot \frac{\partial \mathbf{x}_t^v}{\partial \mathbf{X}_t} \quad (19)$$

- $\frac{\partial \mathbf{x}_t^v}{\partial \mathbf{X}_t}$ similar to gradient of node w.r.t graph (maybe a connection to laplacian)
- By using Graph Diffusion literature;

$$\frac{\partial \mathbf{x}_t^v}{\partial t} = \sum_{v' \in V, v' \neq v} A(v, v') \cdot (\mathbf{x}_t^v - \mathbf{x}_t^{v'}) \equiv A \quad (20)$$

$$\frac{\partial \mathbf{X}_t}{\partial t} = \sum_{v \in V} \sum_{v' \in V, v' \neq v} A(v, v') \cdot (\mathbf{x}_t^v - \mathbf{x}_t^{v'}) \equiv B \quad (21)$$

SDE on Graphs

- Using Eq. 12, Eq. 13 and Eq. 14 in Eq. 11 it gives

$$\nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t) = \sum_{\mathbf{v} \in V} \nabla_{\mathbf{x}_t^v} \log p_t(\mathbf{X}_t) \cdot \frac{A}{B} \tag{22}$$

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

$$\mathbf{g}_t^2 \nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t) = \sum_{\mathbf{v} \in V} \mathbf{g}_t^2 \nabla_{\mathbf{x}_t^v} \log p_t(\mathbf{X}_t) \cdot \frac{A}{B} \tag{23}$$

SDE on Graphs

- Using Eq. 12, Eq. 13 and Eq. 14 in Eq. 11 it gives

$$\nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t) = \sum_{\mathbf{v} \in V} \nabla_{\mathbf{x}_t^v} \log p_t(\mathbf{X}_t) \cdot \frac{A}{B} \tag{22}$$

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

$$\mathbf{g}_t^2 \nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t) = \sum_{\mathbf{v} \in V} \mathbf{g}_t^2 \nabla_{\mathbf{x}_t^v} \log p_t(\mathbf{X}_t) \cdot \frac{A}{B} \tag{23}$$

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

$$d\mathbf{x}_t^v = [\mathbf{f}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v)) - \mathbf{g}_t^2 \nabla_{\mathbf{x}_t^v} \log p_t(\mathbf{X}_t) \cdot \frac{A}{B}] d\tilde{t} + \mathbf{g}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v)) d\tilde{\mathbf{w}} \tag{24}$$

SDE on Graphs

- Now, as we have assumed a factorized of $\mathbf{f}_t(\mathbf{G}_t)$ and $\mathbf{g}_t(\mathbf{G}_t)$ pertaining towards local contributions (local neighbours (L.N.)) as a result $p_t(\mathbf{G}_t)$ will factorize, we can write for a single node as

$$\frac{\partial \log p_t(\mathbf{X}_t)}{\partial \mathbf{x}_t^v} = \frac{\partial \log p_t(\mathbf{x}_t^1, \mathbf{x}_t^2, ...\mathbf{x}_t^2)}{\partial \mathbf{x}_t^v} = \frac{\partial \log p_t(\mathbf{x}_t^v | \mathcal{N}(\mathbf{x}_t^v))}{\partial \mathbf{x}_t^v} \tag{25}$$

SDE on Graphs

- Now, as we have assumed a factorized of $\mathbf{f}_t(\mathbf{G}_t)$ and $\mathbf{g}_t(\mathbf{G}_t)$ pertaining towards local contributions (local neighbours (L.N.)) as a result $p_t(\mathbf{G}_t)$ will factorize, we can write for a single node as

$$\frac{\partial \log p_t(\mathbf{X}_t)}{\partial \mathbf{x}_t^v} = \frac{\partial \log p_t(\mathbf{x}_t^1, \mathbf{x}_t^2, ...\mathbf{x}_t^2)}{\partial \mathbf{x}_t^v} = \frac{\partial \log p_t(\mathbf{x}_t^v | \mathcal{N}(\mathbf{x}_t^v))}{\partial \mathbf{x}_t^v} \tag{25}$$

- It will modify the equation as

$$d\mathbf{x}_t^v = [\mathbf{f}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v)) - \mathbf{g}_t^2 \nabla_{\mathbf{x}_t^v} \log p_t(\mathbf{x}_t^v | \mathcal{N}(\mathbf{x}_t^v)) \cdot \frac{A}{B}] d\tilde{t} + \mathbf{g}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v)) d\tilde{\mathbf{w}} \tag{26}$$

Connection with other variants

- MPNN:
  - Let $vinV$, and neighbours N(v). MPNNs perform a spatial–based convolution on the node v with $\mathbf{u}$ and $\mathbf{m}$ are trainable functions.

$$\mathbf{x}^{(v)}(s+1) = \mathbf{u}\left[\mathbf{x}^{(v)}(s), \sum_{u \in \mathcal{N}(v)} \mathbf{m}\left(\mathbf{x}^{(v)}(s), \mathbf{x}^{(u)}(s)\right)\right] \tag{27}$$

Connection with other variants

- MPNN:
  - Let *vinV*, and neighbours N(v). MPNNs perform a spatial–based convolution on the node v with **u** and **m** are trainable functions.

$$\mathbf{x}^{(v)}(s+1) = \mathbf{u}\left[\mathbf{x}^{(v)}(s), \sum_{u \in \mathcal{N}(v)} \mathbf{m}\left(\mathbf{x}^{(v)}(s), \mathbf{x}^{(u)}(s)\right)\right] \tag{27}$$

  - For clarity of exposition, let $u(x, y)x + g(y)$ where g is the actual parametrized function, then equation becomes

$$\mathbf{x}^{(v)}(s+1) = \mathbf{x}^{(v)}(s) + \mathbf{g}\left[\sum_{u \in \mathcal{N}(v)} \mathbf{m}\left(\mathbf{x}^{(v)}(s), \mathbf{x}^{(u)}(s)\right)\right] \tag{28}$$

Connection with other variants

- MPNN:
  - Let *vinV*, and neighbours N(v). MPNNs perform a spatial–based convolution on the node v with **u** and **m** are trainable functions.

$$\mathbf{x}^{(v)}(s+1) = \mathbf{u}\left[\mathbf{x}^{(v)}(s), \sum_{u \in \mathcal{N}(v)} \mathbf{m}\left(\mathbf{x}^{(v)}(s), \mathbf{x}^{(u)}(s)\right)\right] \tag{27}$$

  - For clarity of exposition, let $u(x, y)x + g(y)$ where g is the actual parametrized function, then equation becomes

$$\mathbf{x}^{(v)}(s+1) = \mathbf{x}^{(v)}(s) + \mathbf{g}\left[\sum_{u \in \mathcal{N}(v)} \mathbf{m}\left(\mathbf{x}^{(v)}(s), \mathbf{x}^{(u)}(s)\right)\right] \tag{28}$$

  - By Eq. 27,

$$\mathbf{x}_{t+1}^{v} = \mathbf{x}_t^{v} + [\mathbf{f}_t(\mathbf{x}_t^{v}, \mathcal{N}(\mathbf{x}_t^{v})) - \mathbf{g}_t^2 \nabla_{\mathbf{x}_t^{v}} \log p_t(\mathbf{x}_t^{v} | \mathcal{N}(\mathbf{x}_t^{v})) \cdot \frac{A}{B}] d\tilde{t} + \mathbf{g}_t(\mathbf{x}_t^{v}, \mathcal{N}(\mathbf{x}_t^{v})) d\tilde{\mathbf{w}} \tag{29}$$

Connection with other variants

- MPNN:
  - By Eq. 27,

$$\mathbf{f}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v)) \equiv \text{Aggregate the features from local neighbourhood}$$

$$\mathbf{g}_t^2 \nabla_{\mathbf{x}_t^v} \log p_t(\mathbf{x}_t^v | \mathcal{N}(\mathbf{x}_t^v)) \cdot \frac{A}{B} \equiv \text{Regularizing the features w.r.t whole graph structure}$$

Connection with other variants

- MPNN:
  - By Eq. 27,

$$\mathbf{f}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v)) \equiv \text{Aggregate the features from local neighbourhood}$$

$$\mathbf{g}_t^2 \nabla_{\mathbf{x}_t^v} \log p_t(\mathbf{x}_t^v | \mathcal{N}(\mathbf{x}_t^v)) \cdot \frac{A}{B} \equiv \text{Regularizing the features w.r.t whole graph structure}$$

- Similarly can be extended to Graph Attention Networks, Convolutional etc.

Optimize

- For score matching [Hyvärinen 2005], lets define $\psi(\mathbf{X}_t, \theta)$ the model density and likewise $\psi_D(\mathbf{X})$ the score function of distribution of observed data D.

$$\psi(\mathbf{X}_t, \theta) = \sum_{\mathbf{v} \in V} \nabla_{\mathbf{x}_t^v} \log p_t(\mathbf{x}_t^v | \mathcal{N}(\mathbf{x}_t^v)) \cdot \frac{A}{B} \tag{30}$$

Optimize

- For score matching [Hyvärinen 2005], lets define $\psi(\mathbf{X}_t, \theta)$ the model density and likewise $\psi_D(\mathbf{X})$ the score function of distribution of observed data D.

$$\psi(\mathbf{X}_t, \theta) = \sum_{\mathbf{v} \in V} \nabla_{\mathbf{x}_t^v} \log p_t(\mathbf{x}_t^v | \mathcal{N}(\mathbf{x}_t^v)) \cdot \frac{A}{B} \tag{30}$$

- We can use the expected square distance as

$$J(\theta) = \int_{\mathbf{X}_t} p_D(\mathbf{X}) \|\psi(\mathbf{X}_t, \theta) - \psi_D(\mathbf{X}_t)\| d\mathbf{X}_t \approx \mathbb{E}_{\mathbf{X}_t} \|\psi(\mathbf{X}_t, \theta) - \psi_D(\mathbf{X})\| \tag{31}$$

$$\hat{\theta} = \text{argmin}_\theta J(\theta) \tag{32}$$

Optimize

- For score matching [Hyvärinen 2005], lets define $\psi(\mathbf{X}_t, \theta)$ the model density and likewise $\psi_D(\mathbf{X})$ the score function of distribution of observed data D.

$$\psi(\mathbf{X}_t, \theta) = \sum_{\mathbf{v} \in V} \nabla_{\mathbf{x}_t^v} \log p_t(\mathbf{x}_t^v | \mathcal{N}(\mathbf{x}_t^v)) \cdot \frac{A}{B} \tag{30}$$

- We can use the expected square distance as

$$J(\theta) = \int_{\mathbf{X}_t} p_D(\mathbf{X}) \|\psi(\mathbf{X}_t, \theta) - \psi_D(\mathbf{X}_t)\| d\mathbf{X}_t \approx \mathbb{E}_{\mathbf{X}_t} \|\psi(\mathbf{X}_t, \theta) - \psi_D(\mathbf{X})\| \tag{31}$$

$$\hat{\theta} = \text{argmin}_\theta J(\theta) \tag{32}$$

- Also, be written as (using integration by steps)

$$J(\theta) = \int_{\mathbf{X}_t} p_D(\mathbf{X}) \sum_{\mathbf{v} \in V} \left[\partial_v \psi(\mathbf{X}_t, \theta) + \frac{1}{2} \psi(\mathbf{X}_t, \theta)^2 \right] \tag{33}$$

THANK YOU
FEEDBACK?