Updates and Progress

Yogesh Verma
Doctoral Candidate
Aalto University

Updates

- Paper read: 52/1280
  - Attending ICLR'22
  - Evaluating generalization in Gflow Nets for molecule design
  - An auto regressive flow model for 3D molecular geometry generation from scratch
  - Evolving-Graph Gaussian Processes
  - End-to-End Differentiable Physics for Learning and Control [Reading]
  - A 3D Molecule Generative Model for Structure-Based Drug Design [Reading]

Updates

- Paper read: 52/1280
  - Attending ICLR'22
  - Evaluating generalization in Gflow Nets for molecule design
  - An auto regressive flow model for 3D molecular geometry generation from scratch
  - Evolving-Graph Gaussian Processes
  - End-to-End Differentiable Physics for Learning and Control [Reading]
  - A 3D Molecule Generative Model for Structure-Based Drug Design [Reading]
- CNF for generating valid molecules, coding, debugging.....
- Reversible SDEs for graphs
- Ideas in Stack: Thinking

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 $\mathcal{N}(\mathbf{v})$.

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

- 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

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

Conditions for the differential function?

Plethora of options for f exist, but we must make it follow our bed-rock assumption of locality + other respectful conditions like

- Permutation Invariant
- Translation invariant
- Rotation invariant

Examples: SphereNet, Polar based CNN (ours), EGNN, etc.

SphereNet: Spherical Message passing for 3D GNN

- A 3D graph is represented as a 4-tuple G = (u, V, E, P) , P is the set of 3D Cartesian coordinates that contains 3D spatial information for each node, use SCS to represent it

SphereNet: Spherical Message passing for 3D GNN

- A 3D graph is represented as a 4-tuple G = (u, V, E, P) , P is the set of 3D Cartesian coordinates that contains 3D spatial information for each node, use SCS to represent it
- Perform message passing in spherical coordinate system by defining some reference plane w.r.t each node and determining its relative position by considering distance, angle and torsion.
- Main diff: Inclusion of 3D information

SphereNet: Spherical Message passing for 3D GNN

- A 3D graph is represented as a 4-tuple G = (u, V, E, P) , P is the set of 3D Cartesian coordinates that contains 3D spatial information for each node, use SCS to represent it
- Perform message passing in spherical coordinate system by defining some reference plane w.r.t each node and determining its relative position by considering distance, angle and torsion.
- Main diff: Inclusion of 3D information
- Permutation Invariant: ✓
- Translation invariant: ✓
- Rotation invariant ✓

SphereNet: Spherical Message passing for 3D GNN

- A 3D graph is represented as a 4-tuple G = (u, V, E, P) , P is the set of 3D Cartesian coordinates that contains 3D spatial information for each node, use SCS to represent it
- Perform message passing in spherical coordinate system by defining some reference plane w.r.t each node and determining its relative position by considering distance, angle and torsion.
- Main diff: Inclusion of 3D information
- Permutation Invariant: ✓
- Translation invariant: ✓
- Rotation invariant ✓
- Only for 3D graphs as 2D we cant represent in SCS (needs some constraint)

Polar Based CNN:

- Represent in polar coordinates (2D) = (r,a) and use RBF to represent the distribution in plane as RBF bump, where N is neighbours, polar coordinates, $\sigma_{(r_i,a_i)}$ ball around them

$$p(r,a) = \frac{1}{N}\sum_{i}^{N}\mathcal{N}(\mu = (r_i, a_i), \sigma = \sigma_{(r_i,a_i)}) \tag{4}$$

Polar Based CNN:

- Represent in polar coordinates (2D) = (r,a) and use RBF to represent the distribution in plane as RBF bump, where N is neighbours, polar coordinates, $\sigma_{(r_i,a_i)}$ ball around them

$$p(r,a) = \frac{1}{N}\sum_{i}^{N}\mathcal{N}(\mu = (r_i, a_i), \sigma = \sigma_{(r_i,a_i)}) \tag{4}$$

- Use CNN over it, but one needs coordinates to represent.

Polar Based CNN:

- Represent in polar coordinates (2D) = (r,a) and use RBF to represent the distribution in plane as RBF bump, where N is neighbours, polar coordinates, $\sigma_{(r_i,a_i)}$ ball around them

$$p(r,a) = \frac{1}{N}\sum_{i}^{N}\mathcal{N}(\mu = (r_i, a_i), \sigma = \sigma_{(r_i,a_i)}) \tag{4}$$

- Use CNN over it, but one needs coordinates to represent.
- Alternative, use difference of features between neighbouring nodes and represent them w.r.t bond angle or torsional angle and use CNN over it.
- Permutation Invariant: ✓
- Translation invariant: ✓
- Rotation invariant ✓

Polar Based CNN:

- Represent in polar coordinates (2D) = (r,a) and use RBF to represent the distribution in plane as RBF bump, where N is neighbours, polar coordinates, $\sigma_{(r_i,a_i)}$ ball around them

$$p(r,a) = \frac{1}{N}\sum_{i}^{N}\mathcal{N}(\mu = (r_i, a_i), \sigma = \sigma_{(r_i,a_i)}) \tag{4}$$

- Use CNN over it, but one needs coordinates to represent.
- Alternative, use difference of features between neighbouring nodes and represent them w.r.t bond angle or torsional angle and use CNN over it.
- Permutation Invariant: ✓
- Translation invariant: ✓
- Rotation invariant ✓
- Can be extended to 3D. by 3D polar coordinates

Extensions

- Modular** Junction Trees
  - Molecular representation as a junction tree where clusters are only 20 high frequent unique ring-substructure

Extensions

- Modular** Junction Trees
  - Molecular representation as a junction tree where clusters are only 20 high frequent unique ring-substructure
- 3D molecule generation
  - Add 3D information like torsional angle (instead of Cartesian coordinates)
  - Can be extended while choosing f as SphereNet, etc, Can model invariance and equivariance by f
- Argmax $\Longleftrightarrow$ Softmax

Reversible SDE on Graphs

SDE

An Ito SDE can be written as:

$$d\mathbf{X}_t = \mathbf{f}_t(\mathbf{X}_t)dt + \mathbf{g}_t(\mathbf{X}_t)d\mathbf{w} \tag{5}$$

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:

$$d\mathbf{X}_t = \mathbf{f}_t(\mathbf{X}_t)dt + \mathbf{g}_t(\mathbf{X}_t)d\mathbf{w} \tag{5}$$

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)

$$d\mathbf{X}_t = [\mathbf{f}_t(\mathbf{X}_t) - \mathbf{g}_t^2 \nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t)]d\tilde{t} + \mathbf{g}_t(\mathbf{X}_t)d\tilde{\mathbf{w}} \tag{6}$$

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 $X_T \sim p_T$

$$d\mathbf{X}_t = \mathbf{f}_t(\mathbf{X}_t)dt + \mathbf{g}_t(\mathbf{X}_t)d\mathbf{w} \tag{7}$$

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 $X_T \sim p_T$

$$d\mathbf{X}_t = \mathbf{f}_t(\mathbf{X}_t)dt + \mathbf{g}_t(\mathbf{X}_t)d\mathbf{w} \tag{7}$$

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

$$d\mathbf{X}_t = [\mathbf{f}_t(\mathbf{X}_t) - \mathbf{g}_t^2 \nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t)]d\tilde{t} + \mathbf{g}_t(\mathbf{X}_t)d\tilde{\mathbf{w}} \tag{8}$$

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)

$$\mathbf{f}_t(\mathbf{X}_t) = \texttt{Agg}_{\mathbf{v}\in V}(\mathbf{f}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v))) \tag{9}$$

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)

$$\mathbf{f}_t(\mathbf{X}_t) = \texttt{Agg}_{\mathbf{v}\in V}(\mathbf{f}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v))) \tag{9}$$

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

$$\nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t) = \frac{\partial \log p_t(\mathbf{X}_t)}{\partial \mathbf{X}_t} = \sum_{\mathbf{v}\in V} \left(\frac{\partial \mathbf{X}_t}{\partial \mathbf{x}_t^v}\right)^{-1} \frac{\sum_{v`\in V} \partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v} \tag{10}$$

$$= \sum_{\mathbf{v}\in V} \left(\frac{\partial \mathbf{X}_t}{\partial \mathbf{x}_t^v}\right)^{-1} \cdot \sum_{\substack{v`\in V \\ v`\in v\cup\mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v} \tag{11}$$

SDE on Graphs

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

$$\mathbf{g}_t^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{X}_t) = \sum_{\mathbf{v} \in V} \mathbf{g}_t^2 \left(\frac{\partial \mathbf{X}_t}{\partial \mathbf{x}_t^v} \right)^{-1} \sum_{\substack{v' \in V \\ v' \in v \cup \mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v} \tag{12}$$

SDE on Graphs

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

$$\mathbf{g}_t^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{X}_t) = \sum_{\mathbf{v} \in V} \mathbf{g}_t^2 \left(\frac{\partial \mathbf{X}_t}{\partial \mathbf{x}_t^v} \right)^{-1} \sum_{\substack{v' \in V \\ v' \in v \cup \mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v} \tag{12}$$

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

$$d\mathbf{x}_t^v = [\mathbf{f}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v)) - \mathbf{g}_t^2 \left(\frac{\partial \mathbf{X}_t}{\partial \mathbf{x}_t^v} \right)^{-1} \sum_{\substack{v' \in V \\ v' \in v \cup \mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v}] d\tilde{t} + \mathbf{g}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v)) d\tilde{\mathbf{w}} \tag{13}$$

How to do?

- Model the score when training with score matching
- Sampling with score-based MCMC like Langevin MCMC (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 $\left(\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_t^v}\right)^{-1} \sum_{\substack{v' \in V \\ v' \in v \cup \mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v}$ for node v (Can also use some network to approximate $\left(\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_t^v}\right)^{-1}$)

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 $\left(\frac{\partial \mathbf{X}_t}{\partial \mathbf{x}_t^v}\right)^{-1} \sum_{\substack{v' \in V \\ v' \in v \cup \mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v}$ for node v (Can also use some network to approximate $\left(\frac{\partial \mathbf{X}_t}{\partial \mathbf{x}_t^v}\right)^{-1}$)
- 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 $\left(\frac{\partial \mathbf{X}_t}{\partial \mathbf{x}_t^v}\right)^{-1} \sum_{\substack{v' \in V \\ v' \in v \cup \mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v}$ for node v (Can also use some network to approximate $\left(\frac{\partial \mathbf{X}_t}{\partial \mathbf{x}_t^v}\right)^{-1}$)
- Plus points:
  - Less dimensionality of the score conditional $\rightarrow$ easy to accurately approximate in contrast to currently used techniques on images
  - The term $\left(\frac{\partial \mathbf{X}_t}{\partial \mathbf{x}_t^v}\right)^{-1}$ encodes some structure into the score function and also in Langevin MCMC sampling giving structure constrained sampling

$$\mathbf{x}_{i+1}^v \leftarrow \mathbf{x}_i^v + \epsilon \left(\frac{\partial \mathbf{X}_t}{\partial \mathbf{x}_t^v}\right)^{-1} \sum_{\substack{v' \in V \\ v' \in v \cup \mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v} + \sqrt{2\epsilon}\mathbf{z}_i \tag{14}$$

Can we do something about $\left(\frac{\partial \mathbf{X}_t}{\partial \mathbf{x}_t^v}\right)^{-1}$

- Denotes the change in node features w.r.t to change in one node features

Can we do something about $\left(\frac{\partial \mathbf{X}_t}{\partial \mathbf{x}_t^v}\right)^{-1}$

- Denotes the change in node features w.r.t to change in one node features
- Can be approximated considering strong local effects? (chain rule as)

$$\frac{\partial \mathbf{X}_t(\mathbf{x}_t^1, \mathbf{x}_t^2, \ldots, \mathbf{x}_t^V)}{\partial \mathbf{x}_t^v} = \sum_{\substack{v' \in V \\ v' \in v \cup \mathcal{N}(v)}} \frac{\partial \mathbf{X}_t(\mathbf{x}_t^1, \mathbf{x}_t^2, \ldots, \mathbf{x}_t^V)}{\partial \mathbf{x}_t^{v'}} \cdot \frac{\partial \mathbf{x}_t^{v'}}{\partial \mathbf{x}_t^v} \tag{15}$$

Can we do something about $\left(\frac{\partial \mathbf{X}_t}{\partial \mathbf{x}_t^v}\right)^{-1}$

- Denotes the change in node features w.r.t to change in one node features
- Can be approximated considering strong local effects? (chain rule as)

$$\frac{\partial \mathbf{X}_t(\mathbf{x}_t^1, \mathbf{x}_t^2, \ldots, \mathbf{x}_t^V)}{\partial \mathbf{x}_t^v} = \sum_{\substack{v' \in V \\ v' \in v \cup \mathcal{N}(v)}} \frac{\partial \mathbf{X}_t(\mathbf{x}_t^1, \mathbf{x}_t^2, \ldots, \mathbf{x}_t^V)}{\partial \mathbf{x}_t^{v'}} \cdot \frac{\partial \mathbf{x}_t^{v'}}{\partial \mathbf{x}_t^v} \tag{15}$$

- Can be incorporated inside eq.13 as

$$\mathbf{x}_{i+1}^v \leftarrow \mathbf{x}_i^v + \epsilon \sum_{\substack{v' \in V \\ v' \in v \cup \mathcal{N}(v)}} \frac{\partial \mathbf{X}_t(\mathbf{x}_t^1, \mathbf{x}_t^2, \ldots, \mathbf{x}_t^V)}{\partial \mathbf{x}_t^{v'}} \cdot \frac{\partial \mathbf{x}_t^{v'}}{\partial \mathbf{x}_t^v} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v} + \sqrt{2\epsilon}\mathbf{z}_i \tag{16}$$

THANK YOU
FEEDBACK?