Updates and Progress

Yogesh Verma
Doctoral Candidate
Aalto University

Updates

- Paper read: 49/1280
  - Conditional Random fields (https://people.cs.umass.edu/ mccallum/papers/crf-tutorial.pdf)
  - Learning 3D representations of molecular chirality with invariance to bond rotations
  - Pre-training molecular graph representation with 3D geometry
  - PDE-GCN: Novel Architectures for Graph Neural Networks Motivated by Partial Differential Equations
  - Evaluating generalization in Gflow Nets for molecule design [Reading]
  - An auto regressive flow model for 3D molecular geometry generation from scratch[Reading]

Updates

- Paper read: 49/1280
  - Conditional Random fields (https://people.cs.umass.edu/ mccallum/papers/crf-tutorial.pdf)
  - Learning 3D representations of molecular chirality with invariance to bond rotations
  - Pre-training molecular graph representation with 3D geometry
  - PDE-GCN: Novel Architectures for Graph Neural Networks Motivated by Partial Differential Equations
  - Evaluating generalization in Gflow Nets for molecule design [Reading]
  - An auto regressive flow model for 3D molecular geometry generation from scratch[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 $\mathcal{N}(\mathbf{v})$.

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

- 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 **V** and its features **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 **V** and its features **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

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

Model Stack

Model stack (only present the best model in paper)

1. NN diff, argmax Cat [motivational example] ✓
2. GCN, argmax Cat [motivational example] ✓
3. GCN, softmax Cat [Many invalid molecules]
4. GCN, softmax CRF
5. ... + JT
6. 3D GCN, softmax CRF

Surrogate Model for categorical representation

- Objective: Represent $p(\phi_T \mid G)$ which is just the posterior distribution
  - Make a surrogate model by assuming a Dirichlet prior over $\mathrm{Cat}(\phi_\mathrm{T}) \sim \mathrm{Dir}(\alpha_\mathrm{v})$ which gives, where α_v are pseudo counts over node label

$$p(\phi_T|G) = \prod_{v \in G} \mathrm{Dir}(\alpha_v + \mathrm{one\_hot}(v)) \tag{4}$$

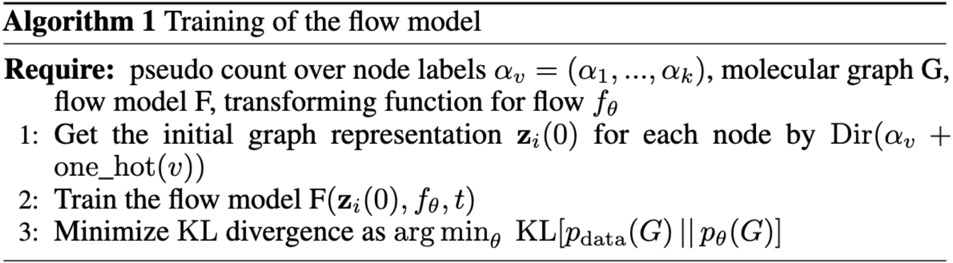
Surrogate Model for categorical representation

- Objective: Represent $p(\phi_T \mid G)$ which is just the posterior distribution
  - Make a surrogate model by assuming a Dirichlet prior over $\mathrm{Cat}(\phi_\mathrm{T}) \sim \mathrm{Dir}(\alpha_\mathrm{v})$ which gives, where α_v are pseudo counts over node label

$$p(\phi_T|G) = \prod_{v \in G} \mathrm{Dir}(\alpha_v + \mathrm{one\_hot}(v)) \tag{4}$$

  - Can be extended to posterior with MC averaging or importance sampling (Anirudh has written in Overleaf link)
- Use this representation to train the flow model instead of dirac peaks.

Training using Surrogate models



Algorithm 1 Training of the flow model

```
Require: pseudo count over node labels $\alpha_v = (\alpha_1, ..., \alpha_k)$, molecular graph G, flow model F, transforming function for flow $f_\theta$
Get the initial graph representation $\mathbf{z}_i(0)$ for each node by $\mathrm{Dir}(\alpha_v + \mathrm{one\_hot}(v))$
Train the flow model $\mathrm{F}(\mathbf{z}_i(0), f_\theta, t)$
Minimize KL divergence as $\arg\min_\theta \ \mathrm{KL}[p_{\mathrm{data}}(G) \,||\, p_\theta(G)]$
```

Conditional Random Fields (CRF) for dependent sampling

- CRFs are an extension to HMM and MRFs which models the conditional distribution $p(\mathbf{y}|\mathbf{x})$ between the state ($\mathbf{y}$) and the observation space ($\mathbf{x}$).

Conditional Random Fields (CRF) for dependent sampling

- CRFs are an extension to HMM and MRFs which models the conditional distribution $p(\mathbf{y}|\mathbf{x})$ between the state $(\mathbf{y})$ and the observation space $(\mathbf{x})$.
- Let G be a factor graph over Y . Then $p(\mathbf{y} \mid \mathbf{x})$ is a conditional random field if for any fixed $\mathbf{x}$, the distribution $p(\mathbf{y} \mid \mathbf{x})$ factorizes according to G. $F = \{\Psi_A\}$ is the set of factors in G, and each factor takes the exponential family form, distribution can be written as

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{\Psi_A \in G} \exp \left\{ \sum_{k=1}^{K(A)} \lambda_{Ak} f_{Ak} \left(\mathbf{y}_A, \mathbf{x}_A \right) \right\} \tag{5}$$

Conditional Random Fields (CRF) for dependent sampling

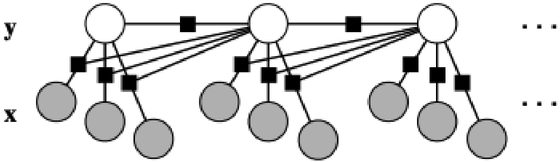
- CRFs are an extension to HMM and MRFs which models the conditional distribution $p(\mathbf{y}|\mathbf{x})$ between the state $(\mathbf{y})$ and the observation space $(\mathbf{x})$.
- Let G be a factor graph over Y . Then $p(\mathbf{y} \mid \mathbf{x})$ is a conditional random field if for any fixed $\mathbf{x}$, the distribution $p(\mathbf{y} \mid \mathbf{x})$ factorizes according to G. $F = \{\Psi_A\}$ is the set of factors in G, and each factor takes the exponential family form, distribution can be written as

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{\Psi_A \in G} \exp \left\{ \sum_{k=1}^{K(A)} \lambda_{Ak} f_{Ak} \left(\mathbf{y}_A, \mathbf{x}_A \right) \right\} \tag{5}$$

- λ_{Ak} and f_{Ak} are parameters and feature functions, which can be shared across whole CRF leading to linear chains CRF as

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp \left\{ \sum_{k=1}^{K} \lambda_k f_k \left(y_t, y_{t-1}, \mathbf{x}_t \right) \right\} \tag{6}$$

Conditional Random Fields (CRF) for dependent sampling



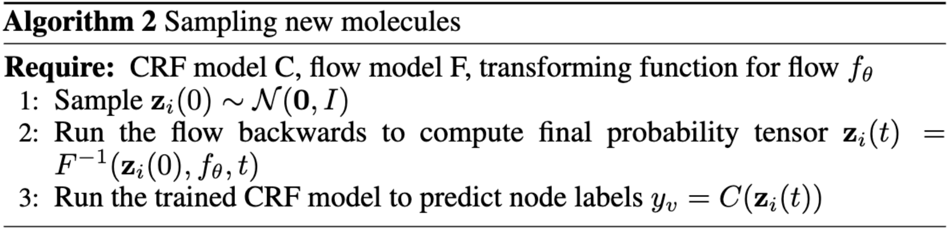
Conditional Random Fields (CRF) for dependent sampling

- Training:
  - Use the surrogate model (Dirchilet Dist.) to get the probability tensor representation of the graph
  - Use that graph to train CRFs model conditional distribution over factor graph defined by neighborhood of each node (modifying x_t) and state space are node labels

Conditional Random Fields (CRF) for dependent sampling

- Training:
  - Use the surrogate model (Dirchilet Dist.) to get the probability tensor representation of the graph
  - Use that graph to train CRFs model conditional distribution over factor graph defined by neighborhood of each node (modifying x_t) and state space are node labels
- Sampling:
  - We have the final probability tensor by Flow model.
  - Use the trained CRF to predict the node-labels of the molecular graph (ordering may have an effect here as it is a sequence model, but neeighbourhood has a say here).

Sampling with CRF and Flow based models



Algorithm 2 Sampling new molecules

```
Require: CRF model C, flow model F, transforming function for flow $f_\theta$
Sample $\mathbf{z}_i(0) \sim \mathcal{N}(\mathbf{0}, I)$
Run the flow backwards to compute final probability tensor $\mathbf{z}_i(t) = F^{-1}(\mathbf{z}_i(0), f_\theta, t)$
Run the trained CRF model to predict node labels $y_v = C(\mathbf{z}_i(t))$
```

Reversible SDE on Graphs

SDE

An Ito SDE can be written as:

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

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:

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

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)

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

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 $X_T \sim p_T$

$$d\mathbf{X}_t = \mathbf{f}_t(\mathbf{X}_t)dt + \mathbf{g}_t(\mathbf{X}_t)d\mathbf{w} \tag{9}$$

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 $X_T \sim p_T$

$$d\mathbf{X}_t = \mathbf{f}_t(\mathbf{X}_t)dt + \mathbf{g}_t(\mathbf{X}_t)d\mathbf{w} \tag{9}$$

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

$$d\mathbf{X}_t = [\mathbf{f}_t(\mathbf{X}_t) - \mathbf{g}_t^2 \nabla_{\mathbf{X}_t} \mathrm{log} p_t(\mathbf{X}_t)]d\tilde{t} + \mathbf{g}_t(\mathbf{X}_t)d\tilde{\mathbf{w}} \tag{10}$$

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)

$$\mathbf{f}_t(\mathbf{X}_t) = \mathtt{Agg}_{\mathbf{v}\in V}(\mathbf{f}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v))) \tag{11}$$

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)

$$\mathbf{f}_t(\mathbf{X}_t) = \texttt{Agg}_{\mathbf{v}\in V}(\mathbf{f}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v))) \tag{11}$$

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

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

$$= \sum_{\mathbf{v}\in V} \left(\frac{\partial \mathbf{X}_t}{\partial \mathbf{x}_t^v}\right)^{-1} \cdot \sum_{\substack{v`\in V \\ v`\in v \cup \mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v} \tag{13}$$

SDE on Graphs

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

$$\mathbf{g}_t^2 \nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t) = \sum_{\mathbf{v} \in V} \mathbf{g}_t^2 \left(\frac{\partial \mathbf{X}_t}{\partial \mathbf{x}_t^v} \right)^{-1} \sum_{\substack{v` \in V \\ v` \in v \cup \mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v} \tag{14}$$

SDE on Graphs

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

$$\mathbf{g}_t^2 \nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t) = \sum_{\mathbf{v}\in V} \mathbf{g}_t^2 \left(\frac{\partial \mathbf{X}_t}{\partial \mathbf{x}_t^v}\right)^{-1} \sum_{\substack{v`\in V\\ v`\in v\cup\mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'}|\mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v} \tag{14}$$

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

$$d\mathbf{x}_t^v = [\mathbf{f}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v)) - \mathbf{g}_t^2 \left(\frac{\partial \mathbf{X}_t}{\partial \mathbf{x}_t^v}\right)^{-1} \sum_{\substack{v`\in V\\ v`\in v\cup\mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'}|\mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v}] d\tilde{t} + \mathbf{g}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v)) d\tilde{\mathbf{w}} \tag{15}$$

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 → 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 → 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

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

THANK YOU

FEEDBACK?