Updates and Progress

Yogesh Verma
Doctoral Candidate
Aalto University

Updates

- Paper read: 27/1280
  - Spanning Tree-based Graph Generation for Molecules
  - Diffusion Kernels on Graphs and Other Discrete Input Spaces
  - Molecular Surface Representation Using 3D Zernike Descriptors for Protein Shape Comparison and Docking
  - Learning Continuous-time PDEs from sparse data with GNNs
  - Energy-Inspired Molecular Conformation Optimization [Reading]
  - Chemical-Reaction-Aware Molecule Representation Learning [Reading]

Updates

- Paper read: 27/1280
  - Spanning Tree-based Graph Generation for Molecules
  - Diffusion Kernels on Graphs and Other Discrete Input Spaces
  - Molecular Surface Representation Using 3D Zernike Descriptors for Protein Shape Comparison and Docking
  - Learning Continuous-time PDEs from sparse data with GNNs
  - Energy-Inspired Molecular Conformation Optimization [Reading]
  - Chemical-Reaction-Aware Molecule Representation Learning [Reading]
- CNF for generating valid molecules and implementation (Need f)
- Molecular surface by 3D Zerneike Descriptors

Problem?

- Learn realistic molecular distributions and generating valid molecules

How do we attack it?

- We represent atom configurations by modelling local neighborhoods with coupled PDE CNFs in graph domain
- Attack validity by accurate local densities

Recap

- Given a molecule with a graph representation (connectivity C) $\mathcal{G} = (V, E)$, one can define a probability distribution at each node over the atomic vocabulary (Vocabulary of all the unique atoms spanning the whole molecule data-set) conditional over each node(atom) neighbours.

Recap

- Given a molecule with a graph representation (connectivity C) $\mathcal{G} = (V, E)$, one can define a probability distribution at each node over the atomic vocabulary (Vocabulary of all the unique atoms spanning the whole molecule data-set) conditional over each node(atom) neighbours.

The total distribution for the whole graph can be written as

$$P(X) = \Pi_i^{N(V)} p(x_i | N_{J(i)}) \tag{1}$$

- 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

Flowing Molecular Graphs

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

$$\begin{bmatrix} \dot{\mathbf{log}}(\mathbf{p}(\mathbf{x_1(t)}|\mathbf{N_{J(1)}})) \\ \dot{\mathbf{log}}(\mathbf{p}(\mathbf{x_2(t)}|\mathbf{N_{J(2)}})) \\ \vdots \\ \dot{\mathbf{log}}(\mathbf{p}(\mathbf{x_n(t)}|\mathbf{N_{J(n)}})) \end{bmatrix} = \begin{bmatrix} -Tr(\frac{\partial \mathbf{f(X(t)}, \mathbf{N_{J(1)}}}{\partial \mathbf{x_1(t)}}) \\ -Tr(\frac{\partial \mathbf{f(X(t)}, \mathbf{N_{J(2)}}}{\partial \mathbf{x_2(t)}}) \\ \vdots \\ -Tr(\frac{\partial \mathbf{f(X(t)}, \mathbf{N_{J(n)}}}{\partial \mathbf{x_n(t)}}) \end{bmatrix} \tag{2}$$

where each $\mathbf{x_i(t)}$ is i^{th} node, $\mathbf{N_{J(i)}}$ its neighbours and $\mathbf{x_i} \in \mathbf{X}$.

Defining f

We define a specific choice of f as

$$f(x_i, N_{J(i)}, \theta = \{\theta_{ij}\}) = \sum_{j}^{N_{J(i)}} \theta_{ij}\phi(x_i, x_j) \tag{3}$$

- where $\phi(x_i, x_j)$ are the radial basis functions and θ_{ij} are the parameters. The task now reduces to learn the distribution of parameters θ_{ij}, given the target distribution on nodes of molecules.

Defining f

We define a specific choice of f as

$$f(x_i, N_{J(i)}, \theta = \{\theta_{ij}\}) = \sum_{j}^{N_{J(i)}} \theta_{ij}\phi(x_i, x_j) \tag{3}$$

- where $\phi(x_i, x_j)$ are the radial basis functions and θ_{ij} are the parameters. The task now reduces to learn the distribution of parameters θ_{ij}, given the target distribution on nodes of molecules.

Constraints:

- Variable neighbours of each atom depending on n-hop neighbours

How to learn θ_{ij}? Functional?

- Similar to Message passing/Attention of GNNs

$$e_{ij} = \text{LeakyRELU}(\vec{a}(x_i||x_j)) \tag{4}$$

$$\theta_{ij} = \frac{exp(e_{ij})}{\sum_{k \in N_{J(i)}} exp(e_{ik})} \tag{5}$$

How to learn θ_{ij}? Functional?

- Similar to Message passing/Attention of GNNs

$$e_{ij} = \text{LeakyRELU}(\vec{a}(x_i||x_j)) \tag{4}$$

$$\theta_{ij} = \frac{exp(e_{ij})}{\sum_{k \in N_{J(i)}} exp(e_{ik})} \tag{5}$$

- RNN or Transformer architecture; shared parameters

$$\theta_{ij} = \text{RNN}(\mathbf{x_i}|N_{J(i)})) \tag{6}$$

Optimization

- Minimize the $D_{KL}\ [p(x(t)|N_J, \theta)\ ||\ p(x^*|N_J)]$ to fit the flow based model, which can be formally written as

$$\mathcal{L}(\theta) = D_{KL}\ [p(x(t)|N_J, \theta)\ ||\ p(x^*|N_J)] \tag{7}$$
$$= \mathbb{E}_{p(x(t)|N_J,\theta)}\ [\log(p(x(t)|N_J, \theta)) - \log(p(x^*|N_J)] \tag{8}$$

- Maximize $p(y_i|x_i(t)) \cdot p(x_i(t))$, where $p(y_i|x_i(t))$ is likelihood defined by Bernoulli which gives cross-entropy

Benefits!

- Non-Auto-regressive or Non-sequential method
- Factorised density
- Rotational invariance (Adjacency is invariant to rotation of graph molecule)
- Approach to generate valid molecules by accurate representation of local structure leads to accurate global structure

A diffusion based approach

A diffusion based approach (Gaussian diffusion models)

- Given $x_0 \sim q(x_0)$, a Markovian noising process q adds noise to the data to produce noised samples $x_{1:T}$ where, each step of the noising process adds Gaussian noise according to some variance schedule given by β_t

$$q(x_t|x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathbf{I}\right) \tag{9}$$

A diffusion based approach (Gaussian diffusion models)

- Given $x_0 \sim q(x_0)$, a Markovian noising process q adds noise to the data to produce noised samples $x_{1:T}$ where, each step of the noising process adds Gaussian noise according to some variance schedule given by β_t

$$q(x_t|x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathbf{I}\right) \tag{9}$$

- One need not apply q repeatedly to sample from $x_t \sim q(x_t|x_0)$. Instead, $q(x_t|x_0)$ can be expressed as a gaussian distribution, with $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \Pi_{s=0}^{t}\alpha_s$

$$q\left(x_t \mid x_0\right) = \mathcal{N}\left(x_t; \sqrt{\bar{\alpha}_t}x_0, \left(1-\bar{\alpha}_t\right)\mathbf{I}\right) \tag{10}$$

$$= \sqrt{\bar{\alpha}_t}x_0 + \epsilon\sqrt{1-\bar{\alpha}_t}, \epsilon \sim \mathcal{N}(0, \mathbf{I}) \tag{11}$$

Here, $1 - \bar{\alpha}_t$ tells us the variance of the noise for an arbitrary timestep.

Reversing the process

- Using Bayes theorem, one finds that the posterior $q(x_{t-1}|x_t, x_0)$ follows gaussian with

$$\tilde{\mu}_t(x_t, x_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t \tag{12}$$

$$\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \tag{13}$$

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}\left(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t \mathbf{I}\right) \tag{14}$$

Reversing the process

- Using Bayes theorem, one finds that the posterior $q(x_{t-1}|x_t, x_0)$ follows gaussian with

$$\tilde{\mu}_t(x_t, x_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t \tag{12}$$

$$\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \tag{13}$$

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}\left(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t \mathbf{I}\right) \tag{14}$$

- If we wish to sample from the data distribution $q(x_0)$ [Data Distribution], we can first sample from $q(x_T)$ and then sample reverse steps $q(x_{t-1}|x_t)$ until we reach x_0. (Unconditional noise reversing)
- All that is left is to approximate $q(x_{t-1}|x_t)$ using a neural network, since it cannot be computed exactly when the data distribution is unknown.

How to find $q(x_{t-1}|x_t)$?

- It has been observed that $q(x_{t-1}|x_t)$ approaches a diagonal Gaussian distribution as $T \to \infty$ and correspondingly $\beta_t \to 0$, so it is sufficient to train a neural network to predict a mean μ_θ and a diagonal covariance matrix Σ_θ

$$p\left(x_{t-1} \mid x_t\right) = \mathcal{N}\left(x_{t-1}; \mu_\theta\left(x_t, t\right), \Sigma_\theta\left(x_t, t\right)\right) \tag{15}$$

Connection to Graphs

- Given a set of vertices X for a graph (molecule), where each $\mathbf{x_i(t)}$ is i^{th} node, $\mathbf{N_{J(i)}}$ its neighbours and $\mathbf{x_i} \in \mathbf{X}$
- We can define a diffusion process similar to Eq.9,10,11 for each $\mathbf{x_i}$

$$q\left(x_0(t) \mid x_0(0)\right) = \mathcal{N}\left(x_0(t); \sqrt{\bar{\alpha}_t} x_0(0), (1 - \bar{\alpha}_t)\,\mathbf{I}\right) \tag{16}$$

$$= \sqrt{\bar{\alpha}_t} x_0(0) + \epsilon\sqrt{1 - \bar{\alpha}_t}, \epsilon \sim \mathcal{N}(0, \mathbf{I}) \tag{17}$$

Conditional Reversing Process

- To condition on variable y, it suffices to sample each transition as

$$p_{\theta,\phi}(x_i(t)|x_i(t+1),y) \propto p_\theta(x_i(t)|x_i(t+1))p_\phi(y|x_i(t)) \tag{18}$$

- Represent as,

$$p_\theta\left(x_i(t) \mid x_i(t+1)\right) = \mathcal{N}(\mu, \Sigma) \tag{19}$$

$$\log p_\theta\left(x_i(t) \mid x_i(t+1)\right) = -\frac{1}{2}\left(x_i(t)-\mu\right)^T \Sigma^{-1}\left(x_i(t)-\mu\right) + C \tag{20}$$

Conditional Reversing Process

- We can approximate $\log p_\phi(y|x_i(t))$ using a Taylor expansion around $x_i(t) = \mu$ as

$$\log p_\phi(y \mid x_i(t)) \approx \log p_\phi(y \mid x_i(t))|_{x_i(t)=\mu} + (x_i(t) - \mu) \nabla_{x_i(t)} \log p_\phi(y \mid x_i(t))\big|_{x_i(t)=\mu} \quad (21)$$

$$= (x_i(t) - \mu) g + C_1 \quad (22)$$

Conditional Reversing Process

- We can approximate $\log p_\phi(y|x_i(t))$ using a Taylor expansion around $x_i(t) = \mu$ as

$$\log p_\phi(y \mid x_i(t)) \approx \log p_\phi(y \mid x_i(t))|_{x_i(t)=\mu} + (x_i(t) - \mu) \nabla_{x_i(t)} \log p_\phi(y \mid x_i(t))\big|_{x_i(t)=\mu} \quad (21)$$

$$= (x_i(t) - \mu) g + C_1 \quad (22)$$

$$\begin{aligned}\log\left(p_\theta(x_i(t) \mid x_i(t+1)) p_\phi(y \mid x_i(t))\right) &\approx -\frac{1}{2}(x_i(t) - \mu)^T \Sigma^{-1}(x_i(t) - \mu) + (x_i(t) - \mu) g + C_2 \\ &= -\frac{1}{2}(x_i(t) - \mu - \Sigma g)^T \Sigma^{-1}(x_i(t) - \mu - \Sigma g) \\ &+ \frac{1}{2} g^T \Sigma g + C_2 \\ &= -\frac{1}{2}(x_i(t) - \mu - \Sigma g)^T \Sigma^{-1}(x_i(t) - \mu - \Sigma g) + C_3 \\ &= \log p(z) + C_4, z \sim \mathcal{N}(\mu + \Sigma g, \Sigma)\end{aligned} \quad (23)$$

Conditional Reversing Process

- Now, $x_i(t-1) \sim \mathcal{N}(\mu + \Sigma \nabla_{x_i(t)} \log p_\phi\left(y \mid x_i(t)\right), \Sigma)$

Conditional Reversing Process

- Now, $x_i(t-1) \sim \mathcal{N}(\mu + \Sigma \nabla_{x_i(t)} \log p_\phi \left(y \mid x_i(t)\right), \Sigma)$
- One can extend or construct $\nabla_{x_i(t)} \log p_\phi \left(y \mid x_i(t)\right)$ to a conditional on nearby neighbours of node such that

$$\nabla_{x_i(t)} \log p_\phi \left(y \mid x_i(t)\right) = f_\phi(x_i(t), N_{J(i)}) \tag{24}$$

3D Molecular Surface Generation

Recap: 3D Molecular Surface Generation

- The 3DZD is a mathematical series expansion of 3D function, which project a 3D object to a compact representation.
- These moments are computed as a projection of the function defining the object onto a set of orthonormal functions within the unit ball – the 3D Zernike polynomials
- The 3D Zernike moments of $f(\mathbf{x})$ (conversion to Cartesian coordinates $Z_{nl}^{m}(\mathbf{x})$ using harmonic polynomials) are defined as the coefficients of the expansion in this orthonormal basis by

$$\Omega_{nl}^{m} = \frac{3}{4\pi} \int_{|\mathbf{x}|\leq 1} f(\mathbf{x}) \bar{Z}_{nl}^{m}(\mathbf{x}) d\mathbf{x} \tag{25}$$

$$\frac{3}{4\pi} \int_{\|\mathbf{x}\|\leq 1} Z_{nl}^{m}(\mathbf{x}) \cdot \overline{Z_{n'l'}^{m'}(\mathbf{x})} d\mathbf{x} = \delta_{nn'} \delta_{ll'} \delta^{mm'} \tag{26}$$

Reconstruction

Since the functions Z_{nl}^m form a complete orthonormal system, it is possible to approximate the original function $f(\mathbf{x})$ by a finite number of 3D Zernike moments (Ω_{nl}^m):

$$\hat{f}(\mathbf{x}) = \sum_n \sum_l \sum_m \Omega_{nl}^m \cdot Z_{nl}^m(\mathbf{x}) \tag{27}$$

- Aim to find these 3D moments and then reconstruct the surface (generative model)

Surface with properties

- One can extend the above formalism to represent various other properties over surface of molecule as 3d function like activity, toxicity, etc.

Surface with properties

- One can extend the above formalism to represent various other properties over surface of molecule as 3d function like activity, toxicity, etc.
- Aim for disentangled representations (variables) for these properties in latent space

Surface with properties

- One can extend the above formalism to represent various other properties over surface of molecule as 3d function like activity, toxicity, etc.
- Aim for disentangled representations (variables) for these properties in latent space
- Need: 3D moments to reconstruct the various surface

GenMol3D

- Given an input point cloud X ($\mathrm{R}^3 \times d$) where d denotes feature vector for each point in that cloud (analogous to local properties of molecule)
- An encoder $Q_\phi(z_d|X)$ infers a posterior over disentangled representations (d) over shape and features (properties)

$$z_d \sim Q_\phi(z_d|X) \tag{28}$$

- Compute the reconstruction likelihood of X (L_{reco}) through CNF G_θ conditioned on z_d. Model can be trained end-to-end to maximize the evidence lower bound (ELBO)

Optimization

$$\log P_\theta(X) \geq \mathrm{E}_{Q_\phi(z_d|X)}[\log P_\theta(X|z_d)] - \mathrm{D}_{KL}(Q_\phi(z_d|X)|P(z_d)) = \mathcal{L}(X;\phi,\theta) \tag{29}$$

where $P_\theta(X|z_d)$ is the decoder and $P_\psi(z_d)$ is prior of latent variables.

$$\log P_\theta(X|z_d) = \sum_{x\in X}\sum_{z\in z_d} \log P_\theta(x|z) \tag{30}$$

$$\tag{31}$$

THANK YOU
FEEDBACK?