Updates and Progress

Yogesh Verma
Doctoral Candidate
Aalto University

Updates

- Paper read: 45/1280
  - Spherical Message Passing for 3D Graph Networks
  - Learning to extend molecular scaffolds with structural motifs
  - Molecular RNN
  - Pre-training molecular graph representation with 3D geometry [Reading]
  - Learning 3D representations of molecular chirality with invariance to bond rotations[Reading]

Updates

- Paper read: 45/1280
  - Spherical Message Passing for 3D Graph Networks
  - Learning to extend molecular scaffolds with structural motifs
  - Molecular RNN
  - Pre-training molecular graph representation with 3D geometry [Reading]
  - Learning 3D representations of molecular chirality with invariance to bond rotations[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 $\mathbf{V}$ 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

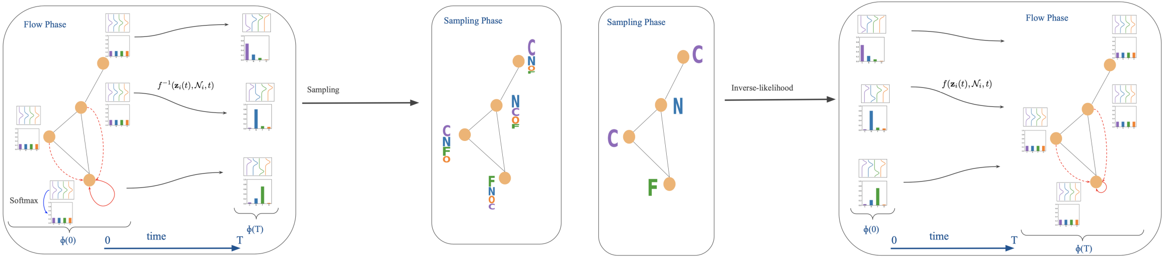
- Given a set of vertices $\mathbf{V}$ and its features $\mathbf{X}$ for a graph (molecule), the goal is to learn the joint distribution $P(G)$ given by Eq.1 .
- For continuous time dynamics of each $\mathbf{v} \in V$, by following Eq. 3,4 we formulate an ODE system as follows

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

- Then the change in log probability follows

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

Workflow



Flow Phase
$f^{-1}(\mathbf{z}_i(t), \mathcal{N}_i, t)$
Softmax
ϕ(0)
0
time
T
ϕ(T)
Sampling
Sampling Phase
Sampling Phase
Inverse-likelihood
Flow Phase
$f(\mathbf{z}_i(t), \mathcal{N}_i, t)$
ϕ(0)
0
time
T
ϕ(T)

Comparison

Table 1: A comparison of the abilities of generative modeling approaches

Method	One-shot	Likelihood	Modular	Invertible	Continuous-time	
JT-VAE	✓	hybrid	✓	✗	✗	Jin et al. (2018)
MRNN	✗	hybrid	✗	✗	✗	Popova et al. (2019)
GraphAF	✗	ARGMAX	✗	✓	✗	Shi et al. (2020)
GraphDF	✗	hybrid	✗	✓	✗	Luo et al. (2021)
MoFlow	✓	ARGMAX	✗	✓	✗	Zang and Wang (2020)
GraphNVP	✓	ARGMAX	✗	✓	✗	Madhawa et al. (2019)
LocalFlow	✓	SOFTMAX	✓	✓	✓	this work

Some Preliminary checks and results

Model	Reconstruction	Validity	Novelty	Diversity
NN	**100%**	**92.14%**	**100%**	**100%**
GCN	**100%**	**96.41%**	**100%**	**100%**
MoFloW	100%	50.3%	100%	99.99%
MRNN	100%	65.3%	99.8%	99.89%
GraphDF	100%	89.03%	99.8%	99.16%

Table: Random generation performance on ZINC250k dataset

Model	Reconstruction	Validity	Novelty	Diversity
NN	**100%**	**95.08%**	**100%**	**100%**
GCN	**100%**	**96.90%**	**100%**	**100%**
GraphNVP	100%	83.1%	58.2%	99.2%
MoFlow	100%	88.96%	96.4%	98.53%
GraphDF	100%	82.67%	98.1%	97.62%

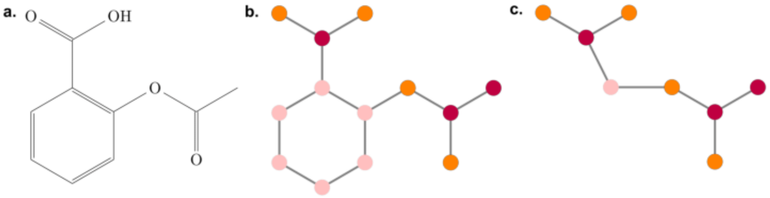
Table: Random generation performance on QM9 dataset

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 ~ 2000 unique rings for 100k ZINC250K data
- But, ~ 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

Some preliminary checks and results

Model	Reconstruction	Validity	Novelty	Diversity
NN(JT)	to do	to do	to do	to do
GCN(JT)	**100**	$\sim$ **94%**	**100%**	**100%**
MoFloW	100%	50.3%	100%	99.99%
MRNN	100%	65.3%	99.8%	99.89%
GraphDF	100%	89.03%	99.8%	99.16%

Table: Random generation performance on ZINC250k dataset using Junction tree approach

Expansions: Cat-sampling

- Generative likelihood $v_{obs} \sim \texttt{Cat}(atom|probs)$ where you evaluate an observed node against you Categorical

$$p(G|\phi) = \prod_{v \in G} \mathrm{Cat}(v|\phi) \tag{4}$$

Expansions: Cat-sampling

- Generative likelihood $v_{obs} \sim \texttt{Cat}(atom|probs)$ where you evaluate an observed node against you Categorical

$$p(G|\phi) = \prod_{v \in G} \mathrm{Cat}(v|\phi) \tag{4}$$

- Inverse likelihood to tranlsate v_{obs} into probbilities $\equiv \mathcal{N}(0.95, 0.3)$

Expansions: Cat-sampling

- Generative likelihood $v_{obs} \sim \texttt{Cat}(atom|probs)$ where you evaluate an observed node against you Categorical

$$p(G|\phi) = \prod_{v \in G} \mathrm{Cat}(v|\phi) \tag{4}$$

- Inverse likelihood to tranlsate v_{obs} into probbilities $\equiv \mathcal{N}(0.95, 0.3)$
- During generation always sample molecules, may lead to many invalid molecules

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

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

SDE on Graphs

- The reverse process can be defined as

$$d\mathbf{X}_t = [\mathbf{f}_t(\mathbf{X}_t) - \mathbf{g}_t^2 \nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t)] d\tilde{t} + \mathbf{g}_t(\mathbf{X}_t) d\tilde{\mathbf{w}} \tag{9}$$

SDE on Graphs

- The reverse process can be defined as

$$d\mathbf{X}_t = [\mathbf{f}_t(\mathbf{X}_t) - \mathbf{g}_t^2 \nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t)] d\tilde{t} + \mathbf{g}_t(\mathbf{X}_t) d\tilde{\mathbf{w}} \tag{9}$$

- 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)

$$\mathbf{f}_t(\mathbf{X}_t) = \mathtt{Agg}_{\mathbf{v} \in V}(\mathbf{f}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v))) \tag{10}$$

SDE on Graphs

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

$$\nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t) = \frac{\partial \log p_t(\mathbf{X}_t)}{\partial \mathbf{X}_t} = \sum_{\mathbf{v} \in V} \frac{\sum_{v' \in V} \partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^{v}} \frac{\partial \mathbf{x}_t^{v}}{\partial \mathbf{X}_t} \tag{11}$$

$$= \sum_{\mathbf{v} \in V} \cdot \sum_{\substack{v' \in V \\ v' = v \text{ or} \\ v' \in \mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^{v}} \frac{\partial \mathbf{x}_t^{v}}{\partial \mathbf{X}_t} \tag{12}$$

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

SDE on Graphs

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

$$\mathbf{g}_t^2 \nabla_{\mathbf{X}_t} \log p_t(\mathbf{X}_t) = \sum_{\mathbf{v} \in V} \mathbf{g}_t^2 \sum_{\substack{v` \in V \\ v` = v \text{ or} \\ v` \in \mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v} \frac{\partial \mathbf{x}_t^v}{\partial \mathbf{X}_t} \tag{13}$$

SDE on Graphs

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

$$\mathbf{g}_t^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{X}_t) = \sum_{\mathbf{v} \in V} \mathbf{g}_t^2 \sum_{\substack{v` \in V \\ v`=v \text{ or} \\ v` \in \mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v} \frac{\partial \mathbf{x}_t^v}{\partial \mathbf{X}_t} \tag{13}$$

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

$$d\mathbf{x}_t^v = [\mathbf{f}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v)) - \mathbf{g}_t^2 \sum_{\substack{v` \in V \\ v`=v \text{ or} \\ v` \in \mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v} \frac{\partial \mathbf{x}_t^v}{\partial \mathbf{X}_t}] d\tilde{t} + \mathbf{g}_t(\mathbf{x}_t^v, \mathcal{N}(\mathbf{x}_t^v)) d\tilde{\mathbf{w}} \tag{14}$$

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.

$$\mathbf{x}^{(v)}(s+1) = \mathbf{u}\left[\mathbf{x}^{(v)}(s), \sum_{u \in \mathcal{N}(v)} \mathbf{m}\left(\mathbf{x}^{(v)}(s), \mathbf{x}^{(u)}(s)\right)\right] \tag{15}$$

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.

$$\mathbf{x}^{(v)}(s+1) = \mathbf{u}\left[\mathbf{x}^{(v)}(s), \sum_{u\in\mathcal{N}(v)} \mathbf{m}\left(\mathbf{x}^{(v)}(s), \mathbf{x}^{(u)}(s)\right)\right] \tag{15}$$

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

$$\mathbf{x}^{(v)}(s+1) = \mathbf{x}^{(v)}(s) + \mathbf{g}\left[\sum_{u\in\mathcal{N}(v)} \mathbf{m}\left(\mathbf{x}^{(v)}(s), \mathbf{x}^{(u)}(s)\right)\right] \tag{16}$$

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.

$$\mathbf{x}^{(v)}(s+1) = \mathbf{u}\left[\mathbf{x}^{(v)}(s), \sum_{u \in \mathcal{N}(v)} \mathbf{m}\left(\mathbf{x}^{(v)}(s), \mathbf{x}^{(u)}(s)\right)\right] \tag{15}$$

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

$$\mathbf{x}^{(v)}(s+1) = \mathbf{x}^{(v)}(s) + \mathbf{g}\left[\sum_{u \in \mathcal{N}(v)} \mathbf{m}\left(\mathbf{x}^{(v)}(s), \mathbf{x}^{(u)}(s)\right)\right] \tag{16}$$

  - By Eq. 27,

$$\mathbf{x}_{t+1}^{v} = \mathbf{x}_t^{v} + [\mathbf{f}_t(\mathbf{x}_t^{v}, \mathcal{N}(\mathbf{x}_t^{v})) - \mathbf{g}_t^2 \sum_{\substack{v' \in V \\ v' = v \text{ or} \\ v' \in \mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^{v}} \frac{\partial \mathbf{x}_t^{v}}{\partial \mathbf{X}_t}] d\tilde{t} + \mathbf{g}_t(\mathbf{x}_t^{v}, \mathcal{N}(\mathbf{x}_t^{v})) d\tilde{\mathbf{w}} \tag{17}$$

Connection with other variants

- MPNN:
  - By Eq. 27,

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

$$\mathbf{g}_t^2 \sum_{\substack{v` \in V \\ v`=v \text{ or} \\ v` \in \mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v} \frac{\partial \mathbf{x}_t^v}{\partial \mathbf{X}_t} \equiv \text{Aggregating the score from its neigbours like MPNN}$$

Connection with other variants

- MPNN:
  - By Eq. 27,

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

$$\mathbf{g}_t^2 \sum_{\substack{v` \in V \\ v`=v \text{ or} \\ v` \in \mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^v} \frac{\partial \mathbf{x}_t^v}{\partial \mathbf{X}_t} \equiv \text{Aggregating the score from its neigbours like MPNN}$$

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

How to do?

- Model the score when training with score matching
- Sampling with score-based MCMC like Langevin MCMC (Similar to multiple noise levels in Song et al. 2020)

Score matching objective

- 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.

$$\psi(\mathbf{X}_t, \theta) = \sum_{\mathbf{v} \in V} \cdot \sum_{\substack{v' \in V \\ v' = v \text{ or} \\ v' \in \mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^{v}} \frac{\partial \mathbf{x}_t^{v}}{\partial \mathbf{X}_t} \tag{18}$$

Score matching objective

- 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.

$$\psi(\mathbf{X}_t, \theta) = \sum_{\mathbf{v} \in V} \cdot \sum_{\substack{v` \in V \\ v`=v \text{ or} \\ v` \in \mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^{v}} \frac{\partial \mathbf{x}_t^{v}}{\partial \mathbf{X}_t} \tag{18}$$

- We can use the expected square distance as

$$J(\theta) = \int_{\mathbf{X}_t} p_D(\mathbf{X}) \| \psi(\mathbf{X}_t, \theta) - \psi_D(\mathbf{X}_t) \| d\mathbf{X}_t \approx \mathbb{E}_{\mathbf{X}_t} \| \psi(\mathbf{X}_t, \theta) - \psi_D(\mathbf{X}) \| \tag{19}$$

$$\hat{\theta} = \text{argmin}_{\theta} J(\theta) \tag{20}$$

Score matching objective

- 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.

$$\psi(\mathbf{X}_t, \theta) = \sum_{\mathbf{v} \in V} \cdot \sum_{\substack{v` \in V \\ v` = v \text{ or} \\ v` \in \mathcal{N}(v)}} \frac{\partial \log p_t(\mathbf{x}_t^{v'} | \mathcal{N}(\mathbf{x}_t^{v'}))}{\partial \mathbf{x}_t^{v}} \frac{\partial \mathbf{x}_t^{v}}{\partial \mathbf{X}_t} \tag{18}$$

- We can use the expected square distance as

$$J(\theta) = \int_{\mathbf{X}_t} p_D(\mathbf{X}) \|\psi(\mathbf{X}_t, \theta) - \psi_D(\mathbf{X}_t)\| d\mathbf{X}_t \approx \mathbb{E}_{\mathbf{X}_t} \|\psi(\mathbf{X}_t, \theta) - \psi_D(\mathbf{X})\| \tag{19}$$

$$\hat{\theta} = \text{argmin}_{\theta} J(\theta) \tag{20}$$

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

$$J(\theta) = \int_{\mathbf{X}_t} p_D(\mathbf{X}) \sum_{\mathbf{v} \in V} \left[\partial_v \psi(\mathbf{X}_t, \theta) + \frac{1}{2} \psi(\mathbf{X}_t, \theta)^2 \right] \tag{21}$$

THANK YOU

FEEDBACK?