Model Outline

Octav Dragoi

June 29, 2020

1 Introduction

The problem is stated as a **molecule optimization** problem: given one molecule, we wish to come up with another one that improves on certain desirable metrics.

The dataset consists of N pairs of molecules from the set of all molecular graphs $\mathcal{G}$, one being the improved version of the other:

$$D = \{(\mathcal{X}_i, \mathcal{Y}_i) : \mathcal{X}_i, \mathcal{Y}_i \in \mathcal{G}, 1 \leq i \leq N\} \tag{1}$$

Our model will employ the following components:

- A **Graph Convolutional Network (GCN)**, learning Wasserstein node embeddings in $\mathbb{R}^d, d \in \mathbb{N}$. This takes the shape of a parametric function G:

  $$G : \mathcal{G} \to \mathscr{D}_2(\mathbb{R}^d), \quad G(\mathcal{X}) \in \mathbb{R}^{|\mathcal{X}| \times d}$$

  where $\mathscr{D}_2(\mathbb{R}^d)$ is the space of all finite point clouds in $\mathbb{R}^d$.

- A nonparametric **tangent space embedding** $\phi = \phi_{Z_0}$ taking a reference point cloud $Z_0 \in \mathbb{R}^{N \times d}$. This function, described in [1], has the form:

  $$\phi : \mathscr{D}_2(\mathbb{R}^d) \to \mathbb{R}^{N \times d}$$

- A pseudoinverse **decoding function** $F \sim G^{-1}$.

  $$F : \mathscr{D}_2(\mathbb{R}^d) \to \mathcal{G}$$

2 Training

Given a graph $\mathcal{X}$, the model should learn an improvement direction $\Delta(\mathcal{X})$. We define this direction as:

$$\Delta(\mathcal{X}_i) = \phi(G(\mathcal{Y}_i)) - \phi(G(\mathcal{X}_i))$$

In other words, we encode the graphs $\mathcal{X}_i, \mathcal{Y}_i$ using the GCN, and then project these encodings onto the tangent space at Z_0. The difference between these tangent space vectors contains the information on how to modify one vector to obtain the other, and this is what the model should learn.

The order of the embeddings within $\mathbb{R}^{N \times d}$ is induced by the order we set on Z_0, and should not affect the overall modeling process.

From [1], $\phi(X) - \phi(Y) \sim W_2(X, Y)$. The quality of this approximation depends on Z_0, and therefore Z_0 should be picked in an appropriate manner to minimize the approximation error.

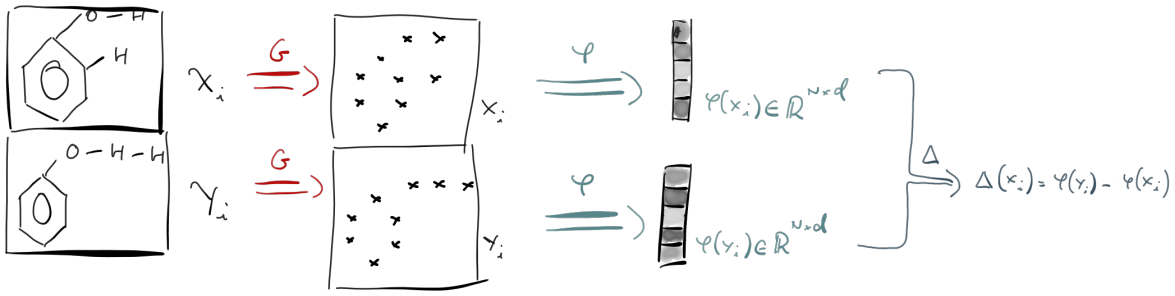

Figure 1: Training flow. Parametrize $\Delta(\mathcal{X}_i)$ in terms of $\mathcal{X}_i$. Only parametric step is the GCN G.

3 Inference

With the information from $\mathcal{X}$ and $\Delta(\mathcal{X})$, the model should be able to decode $\hat{\mathcal{Y}}$. We define the following:

$$\hat{\mathcal{Y}}_i = F\left(\phi^{-1}\left(\Delta(\mathcal{X}_i) + \phi\left(G(\mathcal{X}_i)\right)\right)\right)$$

Referring to [1], ϕ is pseudoinvertible, therefore ϕ^{-1} should be reasonably well defined.

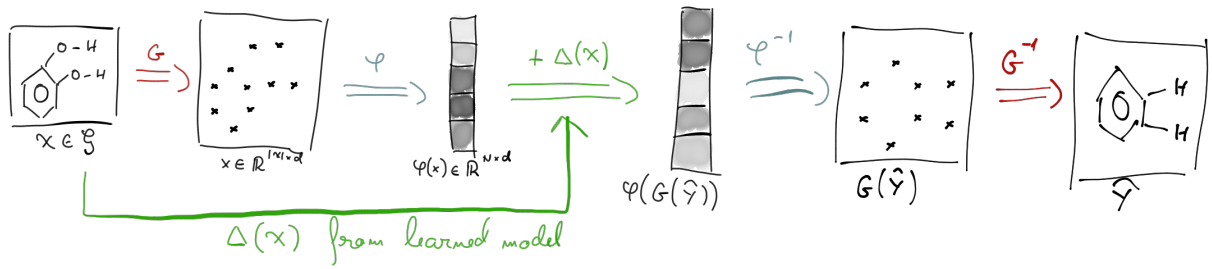

Figure 2: Inference flow. Get the $\Delta(\mathcal{X}_i)$ vector from the trained model, add it to the current embedding and then decode.

4 Open Questions

- The theory around ϕ is scarce. How good is the approximation $\phi(X) - \phi(Y) \sim W_2(X, Y)$? How accurate is the pseudoinverse ϕ^{-1}?
- How to write the decoder F? Pani and I have a few ideas, but nothing too clear for now.
- How to encode the transition from $\mathcal{X}_i$ to $\mathcal{Y}_i$? The difference between tangent vectors is natural, but maybe we can do something fancier with those vectors.

References

[1] S. Kolouri, N. Naderializadeh, G. K. Rohde, and H. Hoffmann, "Wasserstein embedding for graph learning," 2020.