Molecular Geometry Prediction using a Deep Generative Graph Neural Network

Elman Mansimov, Omar Mahmood, Seokho Kang, and Kyunghyun Cho

NYU, Sungkyunkwan University, and Facebook AI Research

Presenter: Eli Draizen

https://qdata.github.io/deep2Read

Outline

Introduction

Computational Drug Design

- Since small molecules/drugs are flexible, **how can we accurately sample their conformational space?**
- When training Protein-Small Molecule Interaction predictors, e.g. with 3DCNNS, is has been shown to be important to include many different conformationd of a ligand bound the same receptor.
  - Hochuli, Helbling, ..., Koes. J Mol Graph Model. 2018

Introduction

Current Approaches

- Exeriental Techniques such as Xray Crystallography
  - Costly and time consuming
- Molecular Force fields / Molecular Dynamics simulations
  - Hand-designed energy functions are crude approximations of the true energy function

figs/md_energy_function.png

Image from https://images.slideplayer.com/23/6581624/slides/slide_2.jpg

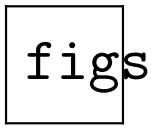
Method

Molecule Representation

- **Input:**
  - Define a molecule as an undirected graph **G=(V, E)**, where the vertices are the atoms and edges represent bonds between them
  - All vertices have vectors of atom features
  - All edges have vectors of bond features
- **Output:**
  - A set of plausible conformations of 3D coordinates
  - $X_a = (x_1^a, ..., x_M^a)$, where $x_i^a \in \mathbb{R}^3$

Method

Atom Features



Method

Bond Features

figs/edge_feats.png

Method

Learn energy function

- Define energy function $\mathcal{F}(X, G)$ of the 3D coordinates and molecule graph, s.t.:
  - $\{X_1, ..., X_S\} = \text{argmin}_X \mathcal{F}(X, G)$
- All S conformations can also be sampled from the Gibbs Distribution
  - $\{X_1, ..., X_S\} \sim p_{\mathcal{F}}(X, G) = \frac{1}{\zeta(G)} exp(-\mathcal{F}(X, G))$
  - ζ is the normalizing constant
- Learn $\mathcal{F}(X, G)$ by maximizing log-likelihood of X^* ground truth conformations
  - $\hat{\mathcal{F}}(X, G) = \text{argmax}_{\mathcal{F}} \frac{1}{N} \sum_{n=1}^{N} log(p_{\mathcal{F}}(X_n^* | G_n))$

Method

Conditional Variational Autoencoder

- Since the log probability, $log(p(X|G))$, is intractable, they maximize its lower bound using a set of latent variables Z:

$$log(p(X|G)) \geq \mathbb{E}_Z(loglikelihood) - KL(posterior|prior)$$
$$\approx \frac{1}{K}\sum_{k=1}^{K} log(p(X|Z^k, G)) - KL(Q(Z|G, X)|P(Z|G)) \quad (1)$$

  - X^k is the k-th sample from approximate posterior
  - P and Q are normal distrributions

Method

Message Passing Neural Networks (MPNN)

- MPNN is Graph NN that operates on the Graph G and is invariant to isomorphisms in G
- For each layer l, they update the hidden vectors for each node ($h(v_i) \in \mathbb{R}^{d_h}$) and hidden matrices for each edge ($h(e_{ij} \in \mathbb{R}^{d_h \times d_h})$) using the equation:

$$h^l(v_i) = GRU(h^{l-1}(vi), J(h^{l-1}(v_i), h^{l-1}(v_{j \neq i}), h(e_{i,j \neq i}))) \quad (2)$$

  - J is is a linear one layer NN that combines info from neighboring nodes via its hidden vectors of nodes and edges
  - GRU is a gated recurrent network that combines the new info from J with the previous layer
  - Weights of J and GRU are shared between layers of the MPNN

Method

Prior Parameterization

- $h^0(v_i)$ and $h(e_{ij})$ are initialized as linear transformations of the feature vectors v_i and e_{ij} of the nodes and edges respectively.
- Final hidden vector passed into 2-layer NN with hidden size d_f, which is transformed into mean, μ_i, and variance, σ_i^2 (with prior weights and biases) of Normal distribution to form the prior distribution:

$$logP(Z|G) = \sum_{i=i}^{N} \sum_{j=1}^{3} -\frac{(\mu_{i,j} - z_{i,j})^2}{2\sigma_{i,j}^2} - log(\sqrt{2\pi\sigma_{i,j}^2}) \qquad (3)$$

- ''Prior distribution as a factorized Normal distribution factored over the vertices and the dimensions in the 3-D coordinate."

Method

Likelihood and Posterior Parameterization

- Likelihood Parameterization
  - ''We incorporate the latent set Z by adding the linear transformation of the node feature vector v_i to its corresponding latent variable z_i''
  - Run neural message passing with new parameters
  - ''The final mean and variance vectors are now three dimensional, representing the 3-D coordinates of each atom, and we can compute the log-probability of the coordinates''
- Posterior Parameterization
  - Input to MPNN is a concatenated edge feature vector e_{ij} and the corresponding distance (proximity) matrix D^* of ground truth conformation X^*
  - Output is a normal distribution for each latent variable
  - Linear weights shared between prior, likelihood, and posterior

Method

Training MPNN

- Before computed log probability, they align all molecules to the mean conformation using Root Mean Square Deviation.
- Regularize Prior by weighting loss function

$$log(p(X|Z^1, G)) - KL(Q(Z|G, X)|P(Z|G)) - \alpha KL(P(Z|G)|P(Z)) \tag{4}$$

  - k=1
  - $\alpha > 0$

Experiment

Datasets

- QM9
  - 133,015 molecules
  - 9 heavy atoms of types C , N , O and F
  - Hold out separate 5,000 and 5,000 randomly selected molecules as validation and test sets.
- COD
  - 66,663 molecules
  - Filtered out any molecule that contains more than 50 heavy atoms of types B, C, N, O, F, Si, P, S, Cl, Ge, As, Se, Br, Te and I.
  - Hold out separate 3,000 and 3,000 randomly selected ones respectively for validation and test purposes.
- CSD
  - 36,985 molecules
  - Filtered out any molecule that contains more than 50 heavy atoms of types B, C, N, O, F, Si, P, S, Cl, Ge, As, Se, Br, Te and I.
  - Hold out separate 3,000 and 3,000 randomly selected ones respectively for validation and test purposes.

Experiment

Datasets

Experiment

Baseline

- Use RDKit to generate conformers using Experimental torsion basic knowledge distance geometry (ETKDG) with two different force fields:
  - Universal Force Field (UFF) – ETKDG+UFF
  - Merck Molecular Force Field (MMFF) – ETKDG+MMFF

Experiment

Model Building

- Hyper parameters set manually and via grid search:
  - Hidden nodes d_h=50 at each layer of all MPNNs
  - Hidden nodes d_f=100 2-layer network after MPNN
  - Layers L=3 for QM9, L=5 for COD and CSD
  - Dropout = 0.2
  - Learning Rate = 3×10^{-4}
  - $\alpha = 10^{-5}$
  - Batch Size = 20 molecules

Results

Overview

- Trained different model for each data set
- Sampled 100 conformations for each molecule in the test set
- They report the median of the mean of the RMSD, the median of the standard deviation of the RMSD and the median of the best (lowest) RMSD among all generated conformations for each test molecule

Results

Number Correct

Figure: Number of successfully processed molecules in the test set, number of successfully generated conformations out of 100, median of mean RMSD, median of standard deviation of RMSD and median of best RMSD per molecule on QM9, COD and CSD datasets.

Results

Computational Efficiency

Results

Best and median RMSDs on COD/CSD as a function of number of heavy atoms

Results

Best and median RMSDs on QM9 as a function of number of heavy atoms

Conclusion

- The Variational Autoencoder generates more diverse conformations than previous methods
- VAE performs faster then baseline even with more atoms
- The model may be improved with more message passage steps and hidden units