Understanding Molecular Data using Graph Convolutional Networks

Suraj Menon

1 Introduction

Using AI to speed up the drug development process is a growing area of research. Drug Discovery is currently a costly and long 10-20 year process. Deep Generative Models, a branch of AI used to generate samples from learned probabilistic distributions, is a promising area that offers interesting research opportunities for quickly generating potential compounds for lab testing.

The goal of this project is to understand how certain deep learning models, specifically Graph Convolutional Networks, are able to learn patterns and structures in datasets of molecules. We use the QM9 dataset containing 133,885 compounds. Within all the different compounds there are five potential atom types (Carbon, Oxygen, etc.) and four different type of chemical bonds between atoms(single bond, double bond, etc.). Each compound has a maximum of 9 atoms within it. We transform these atomic structures into graphs, with the vertices of the graphs being the atoms and the edges being the chemical bonds, and perform a node classification task on the data. As such, the input of the network are vertices and edges of a graph, and the output is a softmax classification of which type of atom each unlabeled vertex is (ie: Carbon).

The eventual goal of having AI models understand molecular data is so that we can train models to generate novel compounds with properties we care about.

2 Related Work

Several interesting projects have tackled this problem before. [1]Zhavoronkov trains a large chemical 'latent' space using a giant database of SMILES (simplified molecular-input line-entry system) molecular data, which it then samples from to generate molecules. It then finds molecules with particular properties that it is looking for by pruning the latent space using Reinforcement Learning. In contrast, [3]De Cao uses a non-likelihood based approach, instead opting for a GAN (Generative Adversarial Network) to generate its compounds instead of a VAE. The project transforms the SMILES molecular data to graphs and then uses a type of GCN (Graphical Convolutional Network) to train a Discriminator, which is then used in an adversarial training model with a Generator to generate likely molecular structures.

An interesting concept we consider as a metric for gauging the value of this work is the idea of 'internal diversity' of generated molecules as introduced by [2]Benhanda. The paper introduces the concept as a way of measuring how 'diverse' the output is of any generative network that is generating compounds in comparison to the set of compounds from which it is trained. It posits the internal diversity I of a set of molecules A of size $|A|$ to be the average of the Tanimoto-distance T_d of molecules of A with respect to each other. We define the internal diversity as:

$$I(A) = \frac{1}{|A|^2} \sum_{(x,y) \in A \times A} T_x(x, y) \tag{1}$$

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

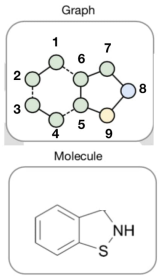
Where the Tanimoto-distance $T_x(x, y)$ is measure of how small of a common 'fingerprint' is there between two molecules.[2] Considering this metric gives us a good conceptual goal of what we would like to achieve with our generative model.

3 Dataset and Features

We use the QM9[4] dataset for training our model. The dataset comes with 133,885 compounds in the SMILES format containing up to five unique atom types and four unique chemical bonds. There are a maximum of 9 number of atoms within any one compound.

The first task is to transform the SMILES data into graphs. To do this, we leverage a code implementation of the MolGAN[3] paper and the RDKit python library to generate a graph representation of the molecules[6]. The output is a one $M \times N$ array X that contains the verteces of the graphs and one $M \times N \times N$ array A that contains the edges. (Here M is the total number of molecules and N is the max number of atoms per molecule, which is 9 in our example.) The values in $X \in (0, Atoms)$ and the values in $A \in (0, Bonds)$ where $Atoms$ is 5 and $Bonds$ is 4. All the values in A are normalized by row before any training begins.

The following figure outlines how a molecule gets put into an array representation before it is fed the model. Consider the vertices numbered as labeled in the figure. From that it is intuitive to follow how the feature vector X and adjacency matrix A for this molecule is created. Treat the atom names as simply colors G, B, and Y in this example. For the adjacency matrix, notice how the value of the double bond in the molecule is labeled 2 in A while a single bond is 1.



$$X = [G, G, G, G, G, G, G, B, Y]$$

$$\begin{aligned}
A_{1,x} &= [0, 1, 0, 0, 0, 2, 0, 0, 0] \\
A_{2,x} &= [1, 0, 2, 0, 0, 0, 0, 0, 0] \\
A_{3,x} &= [0, 2, 0, 1, 0, 0, 0, 0, 0] \\
A_{4,x} &= [0, 0, 1, 0, 2, 0, 0, 0, 0] \\
A_{5,x} &= [0, 0, 0, 2, 0, 1, 0, 0, 1] \\
A_{6,x} &= [2, 0, 0, 0, 1, 0, 1, 0, 0] \\
A_{7,x} &= [0, 0, 0, 0, 0, 1, 0, 1, 0] \\
A_{8,x} &= [0, 0, 0, 0, 0, 0, 1, 0, 1] \\
A_{9,x} &= [0, 0, 0, 0, 1, 0, 0, 1, 0]
\end{aligned}$$

Figure 1: Graph/Molecule Representation into Array Representation

4 Methods

We attempt to measure the understanding of the molecular datasets using a few variations of graph neural networks. We give a summary of our three different methods below:

4.1 Graph Convolutional Networks (GCNs)

The baseline method we used was the Graph Convolutional Network (GCN), using the description and started code from [5]. GCNs at a high level take in an input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as a set of vertices and edges and finds a mapping from the inputs to a set of vectors or 'embeddings' that inherently represent the structural and feature information in the graph. More formally we are trying to find:

$$f : (\mathcal{V}, \mathcal{E}, (x_i^{0,E})_{i \in \mathcal{V}}) \rightarrow z \in \mathbb{R}^{|V| \times d'} \tag{2}$$

Here $x_i^{0,E}$ dictates the feature (for us, the atom type) at vertex i and that it lies in Euclidean space.[4] The output is a set of vectors that represent the original information efficiently in dim d'. We can see the overall pipeline of the GCN for a node classification task in the figure below.

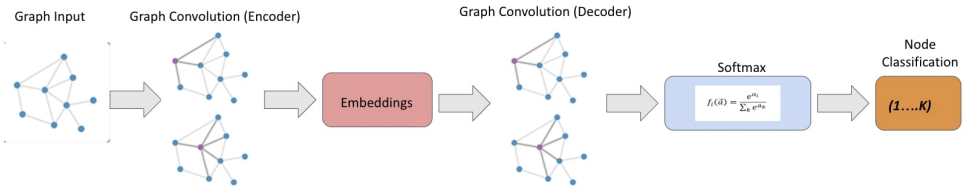


Figure 2: Pipeline of Graph Convolutional Network for Node Classification.[8]

The graph convolution layer is represented by the following equations, where ℓ denotes one layer of the network:

$$h_i^{\ell,E} = W^\ell x_i^{\ell-1,E} + b^\ell \tag{3}$$

$$x_i^{\ell,E} = \sigma(h_i^{\ell,E} + \sum_{j \in \mathcal{N}(i)} w_{ij} h_j^{l,E}) \tag{4}$$

In equation (3), we see that each node feature goes through a linear layer with a shared weight matrix and bias vector among all features to get 'hidden' values for each node. Then, to get the features for the next layer we apply equation (4), which aggregates the hidden values for all nodes that are neighbors of the node in question. In practice, this comes out to a simple matrix multiplication of the matrix of hidden values, which is the output of equation (3), and the adjacency matrix which is a starting input to the algorithm. This layer finishes with a non-linear activation function, which in our case is a ReLU.

Note that this network does not make use of the different edge types. In order to use this information, we try the next slightly more complicated network.

4.2 Relational Graph Convolutional Networks (RGCNs)

In order to leverage the distinction between different bond types in the molecular graphs we also consider using a Relational Graph Convolutional Network (RGCNs).[9] RGCNs are similar to GCNs but assume that different nodes of the graph may be related in different ways and therefore adds parameters in its network to account for different edge types. The equation then from GCNs now transform into:

$$x_i^\ell = \sigma(\sum_r \sum_{j \in \mathcal{N}^r(i)} \frac{1}{c_{i,r}} W_r^\ell h_j^\ell + W_0^\ell h_i^{\ell)}) \tag{5}$$

This equation shows that we get the values for the next layer from aggregating hidden values at each node with specific weight values for each type of edge. Practically, this involves a matrix multiplication of the hidden values from the first linear layer for each different type of edge type (in our case 4, since there are 4 bond types).

4.3 Hyperbolic Graph Convolutional Networks (HGCNs)

A primary hypothesis of this project was to gauge if using hyperbolic embeddings of the molecular data would provide better results in the node classification task, which may prove that it would work as a better base in generative network to produce more accurate and diverse compounds. To test this out, we leveraged with the work from [4] in their work to adapt the GCN to hyperbolic embeddings.

Due to space considerations and necessary background, we will leave it to the reader to find the details of the HGCN in [4]. At a high level, it takes the existing Euclidean features from each node, transforms them into hyperbolic space, and then aggregates its neighbors embeddings in the tangent space of the center node and projects the result to a hyperbolic space with different curvature.

We leverage the library provided in [7] to run the same dataset through a GCN, RGCN, and HGCN and compare our node classification results.

5 Experiments and Results

For all of our experiments we train on a subset of Q9 of 5000 compounds and do a train/val/test split of 80/10/10. Each of the models is trained for node classification and uses a Multinomial Cross Entropy loss function. We judge the performance of our classification task by accuracy. Since false positives are not catastrophic for this task, we primarily care by how well our model can correctly predict the correct classification. For simplicity we keep the number of layers of the network to 2 and the learning rate of .01 as we are aiming less to optimize the metrics but rather to compare different models. We run the models without weight-decay and without any regularization or dropout. We train for a maximum of 300 epochs or until convergence. We generate the train/val/test sets using a randomly shuffling the nodes and using .80 for training, .10 for validation, and .10 for test.

For these results, we are attempting a classification task out of 5 classes, so we would expect an accuracy of .20 for a random selection. For the 5000 compounds uses, we have a total of 17152/2144/2144 nodes in the train/val/test sets respectively.

Model	**Train Accuracy**	**Test Accuracy**
GCN	.99	.57
RGCN	.70	.50
HGCN	.99	.61

Table 1: Evaluation of Accuracy of different models

From these results we can conclude a couple things. We see that for GCN and HGCN, the models are able to totally overfit on the data while on RGCN there are still misclassifications on the train set. We see that this also results in GCN and HGCN giving better results on the test set than RGCN.

This tells us that GCNs are likely well equipped to understand the structure of the molecular data, and likely will give us better results the more that molecules that we train with, and if we add some regularization to the model. Surprisingly RGCN performs worse than GCN, even though it is leveraging the extra bond information in the adjacency matrix. Looking through RGCN's confusion matrix, we see it it has misclassifications between class 0 and class 4 in the training set while GCN does not. A next step would be to analyze why RGCN gives us this misclassification and try to tune our model to fix it. Ideally, we will want our base model in the generative network to leverage the bond type information.

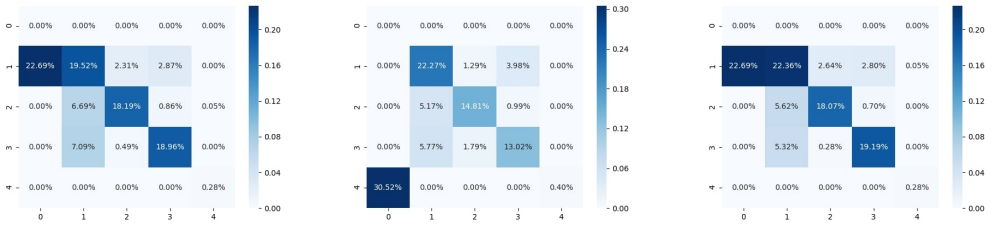


Figure 3: Confusion Matrices. GCN (Left), RGCN (Middle), HGCN (Right)

More pleasingly in line with our hypothesis, we see that HGCN is able to outperform GCN, showing promise that hyperbolic embeddings may indeed offer better performance for understanding these molecular graphs. Training and testing on a larger set of molecules will be able to further confirm this.

6 Conclusions and Further Work

The work done in this project acts as a starting understanding for further investigation in using generative models to aid drug development. From this we learned that GCNs can be used to accurately learn the structure and properties of molecular data. We attempted to fit a RGCN to the

molecular data and realize that the model may need more tuning or a larger training set to correctly fit the data. We also see by testing the data with HGCN that hyperbolic embeddings may provide better results for this task than Euclidean embeddings and so merit further investigation.

To follow up on this work, we should analyze and understand first why RGCN is outperformed by GCN. Once RGCN's performance has improved, we should create a hyperbolic extension to RGCN and see if the performance improves in the same manner that it did from GCN to HGCN.

The next step after that is to use the best performing model as the base for a generative network (such as the Discriminator in a GAN) that learns to generate novel compounds. The first task will be to try to generate back the training set, and then to attempt to maximize the internal diversity(as discussed earlier) of the generated compounds.

Codebase

Leveraging Code from [6] and [7], the code repository can be found here:

https://github.com/sjmickiemouse/cs229_molecules.git

Contributions

Though the project was initially conceived with two members, one member dropped out of the class a month in. All of the work of this project was done by Suraj with the help of the items in the References section.

Thanks to CS229 teaching staff and mentors for their support.

References

[1] Zhavoronkov A & Ivanenkov, Yan a et al. Deep Learning enables rapid identification of potent DDR1 kinase inhibitors. *https://doi.org/10.1038/s41587-019-0224-x*. 2019

[2] Mostapha Benhenda. ChemGAN challenge for drug discovery: Can AI reproduce natural chemical diversity?. *https://arxiv.org/pdf/1708.08227.pdf.* 2017.

[3] Nicola De Cao, Thomas Kipf. MolGAN: An implicit generative model for small molecular graphs. *https://arxiv.org/pdf/1805.11973.pdf*. 2018.

[4] Ramakrishnan, Raghunathan, Dral, Pavlo O, Rupp, Matthias, and Von Lilienfeld, O Anatole. Quantum chem- istry structures and properties of 134 kilo molecules. *Scientific data,* 1:140022, 2014.

[5] Ines Chami, Rex Ying, Christopher Re, Jure Leskovec, Hyperbolic Graph Convolutional Neural Networks. *http://web.stanford.edu/ chami/files/hgcn.pdf* 2019.

[6] MOL-GAN Pytorch implementation. *https://github.com/yongqyu/MolGAN-pytorch*

[7] HGCN Pytorch implementation. *https://github.com/HazyResearch/hgcn*

[8] Thomas Kipf, Graph Convolutional Networks, https://tkipf.github.io/graph-convolutional-networks/. 2017

[9] Michael Schlichtkrull, Thomas N. Kipf a et al. Modeling Relational Data with Graph Convolutional Networks. *https://arxiv.org/pdf/1703.06103.pdf*. 2017