Predicting Molecular Properties with Graph Attention Networks

Rohan Mehrotra
Department of Computer Science
Stanford University
Stanford, CA 94305
rohanm2@stanford.edu

Kevin Guo
Department of Computer Science
Stanford University
Stanford, CA 94305
kyguo@stanford.edu

Abstract

Predicting properties of molecules without the need for lab experiments is a desirable objective that has the potential to revolutionize the development of drugs and other new molecules. Previously, several research groups have worked on this problem, and graph learning methods have emerged as the most promising approach. In this paper, we propose that a recently developed graph learning technique – Graph Attention Networks (GAT's) – could be used to further improve property prediction. We build a GAT model to predict the toxicity of molecules in the Tox21 dataset, and show that it outperforms several baseline models. Furthermore, we look closer at the GAT model's predictions and find that it performs best on smaller, linear molecules, but still has room for improvement on larger, aromatic molecules – a potential area for further research.

1 Introduction

The design of new molecules with particular therapeutic properties is the central goal of the drug development process. Today, biochemists rely on tedious laboratory experiments in order to screen molecules to evaluate their pharmaceutical potential. As a result, the drug development process is both cost-inefficient and time consuming, with a single drug requiring 12 years and $2.6 billion on average to be developed and approved.[1]

Artificial intelligence has the potential to revolutionize the drug development pipeline. There has recently been significant interest in "molecular machine learning": using artificial intelligence to predict molecular properties automatically and efficiently. If the properties of a molecule could be predicted solely based on the molecule's atomic structure, rather than from wet lab experiments, it would represent a major breakthrough in molecule design – with applications in drug discovery and beyond.

In 2014, the NIH's "Toxicology in the 21st Century" (Tox21) initiative published a public dataset reporting the toxicity of a number of compounds.[2] The dataset contains toxicity measurements for 7830 compounds on 12 different target tasks. The NIH released the dataset with the intention of establishing a standard dataset that researchers could use to benchmark models they build for predicting properties based on molecular structures. In this paper, we present a graph attention network-based model for molecular property prediction on the Tox21 dataset. We first review related work, then present our network design, and finally evaluate and rationalize its performance.

2 Related Work

Since the release of Tox21, there have been several reports on the effectiveness of various machine learning approaches on the dataset. Previous groups have developed models based on a number of approaches including logistic regression, random forests, SVM, and graph networks for Tox21, with varying degrees of success.[3, 4, 5]

In 2017, Wu et al.[6] published MoleculeNet, a benchmark designed for testing machine learning methods on molecular properties. MoleculeNet curated the key previously proposed algorithms for Tox21 prediction (and other chemical datasets), and integrated them into an open-source Python package called DeepChem. Out of all the models in MoleculeNet, the most effective one was graph convolutional networks (GCN's). This suggests that graph learning

methods may be the most effective approach to molecular property prediction. Such a result is unsurprising, because molecules are particularly amenable to graph representations. Specifically, molecules can be represented as graphs with nodes representing the atoms and edges representing the bonds between them. The node features and adjacency matrices indicating edges can be fed into graph learning techniques for property prediction.

3 Background on Graph Learning

Graph learning is a quickly evolving field that has seen rapid algorithmic advances in recent years. In 2016, a key breakthrough occurred when Defferrard et al. proposed a "graph convolutional" operation, analogous to convolutions used in CNN's for computer vision.[7] These graph convolutional networks (GCN's) use both node features and topological structural information to make predictions, and have proven to greatly outperform traditional methods for graph learning.

Beyond GCN's, in 2017, Velickovic et al. published a landmark paper introducing attention mechanisms to graph learning, thus proposing introducing a new architecture for graph learning called graph attention networks (GAT's).[8] Through an attention mechanism on neighborhoods, GAT's can more effectively aggregate node information. Recent results have shown that GAT's perform even better than standard GCN's at many graph learning tasks.

In MoleculeNet and other papers, standard GCN's have already been implemented for Tox21 prediction. In this paper, we seek to go beyond standard GCN's by applying the recently developed attention mechanism to Tox21. We hypothesized that, because GAT's learn weights for node aggregation rather than fixing a node aggregation formula, they would be able to more effectively learn local information in molecular structures that hold insights into their overall properties.

4 Dataset

4.1 Overview

The dataset we are using for molecular property prediction is the Tox21 dataset. The Tox21 dataset comprises 7830 chemical compounds. For each sample, there are 12 binary labels (active/inactive) representing the outcome of 12 different toxicological assays. The objective is to build a model that can accurately predict these 12 toxicological properties of molecules based on their structures.

4.2 Featurization

In the NIH's release of Tox21, the molecules are represented as SMILES strings (a standard method of encoding structures). However, because we wanted to use graph learning methods, we needed to convert these SMILES strings to graph representations.

We built our models using two Python libraries: DeepChem (for the baseline models) and DGL (for the GAT model). Both DeepChem and DGL contain methods to transform the SMILES strings into featurized representations that can be fed into graph networks. In particular, these methods convert SMILES strings into graph objects, which consist of an an n-dimensional node feature vector (where n is the number of atoms in the molecule), and a list containing the node pairs $[u, v]$ that are connected with an edge (bond).

5 Methods

5.1 Baseline Models

We first implemented several baseline models using the DeepChem package. In particular, we built a Logistic Regression model, SVM model, and GCN model. DeepChem includes implementations for each of these models that are optimized for Tox21, which we leveraged to build and test the baseline models. The purpose of these baselines was to set a benchmark which we could compare our GAT model's performance to.

5.2 GAT Model

Next, we implemented the GAT model. We decided to build the model using a PyTorch-based library called DGL, which is designed to make the implementation of graph learning methods easier. As a preface, we will first describe the

mathematical intuition behind GAT's, then describe the architecture and hyperparameters that we used in our model for Tox21. We would also like to acknowledge the original GAT paper[8], whose code inspired our own implementation.

The key difference between GAT's and GCN's is how the information from node neighbors is aggregated. For GCN's, the graph convolution produces the normalized sum of the node features of neighbors. GAT introduces the attention mechanism as a substitute for the statically normalized convolution.

As described in the original GAT paper, a GAT layer consists of the following equations to compute the node embedding of layer $l+1$ from the embeddings of layer l:[8]

$$z_i^{(l)} = W^{(l)} h_i^{(l)}, \quad (1)$$

$$e_{ij}^{(l)} = \text{LeakyReLU}(\vec{a}^{(l)^T}(z_i^{(l)} || z_j^{(l)})), \quad (2)$$

$$\alpha_{ij}^{(l)} = \frac{\exp(e_{ij}^{(l)})}{\sum_{k \in \mathcal{N}(i)} \exp(e_{ik}^{(l)})}, \quad (3)$$

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(l)} z_j^{(l)}\right), \quad (4)$$

Equation (1) is a linear transformation of the lower layer embedding $h_i^{(l)}$ and $W^{(l)}$ is its learnable weight matrix. Equation (2) computes a pair-wise un-normalized attention score between two neighbors. Here, it first concatenates the z embeddings of the two nodes, where || denotes concatenation, then takes a dot product of it and a learnable weight vector $a^{(l)}$, and applies a LeakyReLU in the end. Equation (3) applies a softmax to normalize the attention scores on each node's incoming edges. Equation (4) is similar to GCN's: the embeddings from neighbors are aggregated together, scaled by the attention weights.

5.3 Architecture and Hyperparameter Choices

Our GAT model consists of four layers (an input layer, two hidden layers, and an output layer). The hidden layers both consist of 32 hidden units. Finally, we have 12 separate output layers, each consisting of 64 units and outputting a single value, for the 12 separate tasks.

The input layer and hidden layers are GAT layers, while the output layer is a fully connected layer. Intuitively, the GAT layers each accept a node embedding and compute a new node embedding by aggregating local neighborhood information. Finally, the sigmoid activation function is applied to the output of the fully connected layer because each task is a binary classification problem.

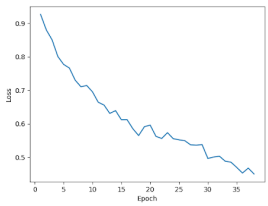
For optimizing weights, we use the Adam optimization algorithm, with a mini-batch size of 128, and a learning rate of 0.001. We trained the model for 100 epochs but with an early stopping mechanism to prevent overfitting. For training and testing, we use an 80/10/10 training/validation/test split of the data.

To evaluate the models, we use the ROC-AUC score ("area under ROC curve") metric. The ROC-AUC score quantifies the diagnostic ability of a binary classifier system. We calculate an ROC-AUC score for each of the 12 tasks, then average over all 12 to calculate an overall score for the model.

6 Experiments/Results/Discussion

6.1 ROC-AUC Scores for Baseline Models and GAT

Model	ROC-AUC
Logistic Regression	0.7129
SVM	0.7672
GCN	0.7810
GAT (Ours)	0.8242



The ROC-AUC scores obtained by each model are shown above. As expected, out of the three baseline models, the GCN performed the best (consistent with previous reports). Our GAT model successfully outperformed the GCN and other baseline models.

The loss per epoch during the training of our GAT model is also shown above. The early stopping mechanism halted training at 38 epochs.

6.2 Architecture and Hyperparameter Choices

When developing our model, we explored several different choices for the architecture and hyperparameters.

We first explored the effect of varying the number of layers. We started with a 3-layer model with just one hidden GAT layer. Then, we built larger models with more GAT layers: a 4 and 5-layer model. We found that the 4-layer model performed best.

A problem we encountered initially was overfitting: we wanted to train for more epochs to improve performance, but after a certain point we overfit. So, we decided to add an early stopping mechanism to counteract this, and it ended up helping with this problem significantly.

Next, we explored varying the learning rate and mini-batch size. A learning rate of 0.001 was consistent with the literature, and we found that varying it did not affect performance much. Next, for the mini-batch size, we found previous Tox21 papers using a size of 128, and found that 128 was indeed a reasonable value for the batch size. So, we stuck with basing these two hyperparameters off of the literature.

6.3 Analyzing the Results of the GAT model

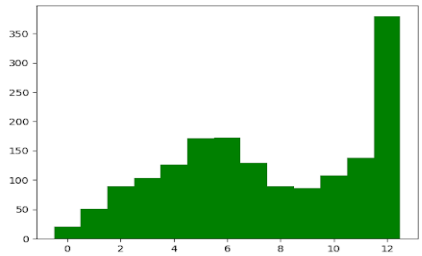
After training the GAT model, we decided to take a deeper dive into the results to understand what the model was learning.

Our first question was: is the model scoring significantly better on certain tasks, out of the 12 total tasks? To evaluate this, we outputted the ROC-AUC score on each task.

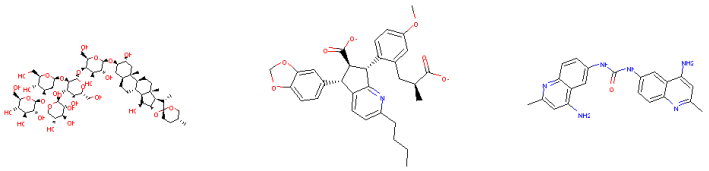
The ROC-AUC score on each task was: [0.6957, 0.8553, 0.8799, 0.8379, 0.7311, 0.8430, 0.8542, 0.7856, 0.8289, 0.8377, 0.9185, 0.8227]. Looking at these scores, although there was natural variability across the tasks, there was no task that the model did significantly better or worse on.

Given that the model was performing similarly on all tasks, we decided to next probe another question: Is the model misclassifying different molecules in each task, or are certain molecules inherently "harder" to classify across all 12 tasks?

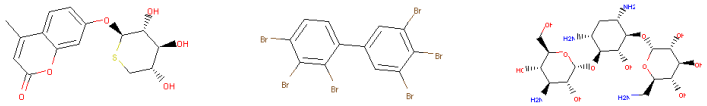
To evaluate this question, we stored the number of tasks (out of 12) that were correctly predicted for each of the 783 molecules in the test set. We then plotted a histogram distribution of the frequency of tasks correctly predicted, which is shown below. The buckets indicate the number of tasks out of 12 correctly predicted. We can see that the model predicted 12/12 tasks (all tasks right!) for 360 molecules, 11/12 correct for 140 molecules, and so on.



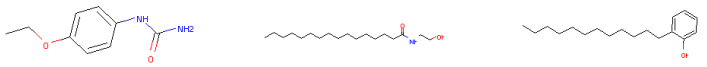
Interestingly, the distribution appears to be bimodal. There is a cluster of molecules that the model predicted very accurately (in the 12/12 bucket), while the rest of the distribution appears roughly normal. To gain more insight into what kinds of molecules the model is doing well on, we randomly sampled and printed a few molecules from the 3/12 bucket, 6/12 bucket, 9/12 bucket, and 12/12 bucket. Three examples for each are shown below.



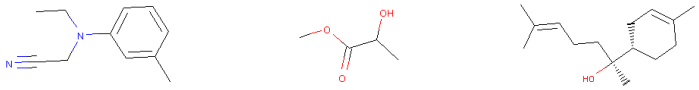
3/12 tasks correct:



6/12 tasks correct:



9/12 tasks correct:



12/12 tasks correct:

The sample molecules suggest that the model performs very well on molecules that are small and linear, but struggles to predict properties for molecules that are larger and have many aromatic rings.

These results are reasonable given how graph learning algorithms work and given the atomic basis of molecular properties. GCN's are good at learning and aggregating information from local neighborhoods, and the attention mechanism makes the model even better at learning complex neighborhood structure. In chemistry, a molecule's properties are often based on certain local motifs (functional groups) that have particular reactivity. For this reason, it makes sense that graph convolutions perform well at predicting properties: as they can capture the structure of local motifs.

For small, linear molecules, local motif information appears to be sufficient for property prediction. However, this effectiveness seems to break down for large molecules with many aromatic rings. Aromatic rings comprise a conjugated system with an appropriate number of electrons. In general, they have unexpected effects on molecules' stability and reactivity.[9] This may be the reason why our GAT model performs poorly on large molecules with many rings: aromaticity causes unexpected effects that "confuse" the model. With aromaticity, local information appears to not lead to the expected property.

7 Future Work

In future work, we would like to modify our model to solve its shortcomings on predicting toxicity for molecules with large, aromatic rings. The attention mechanism endows us with the flexibility to modify the amount of attention we give to certain nodes. In the future, we propose adding a feature that is activated whenever the local environment is part of an aromatic ring. By adding in this feature, the model may be able to better learn the unexpected effects that aromatic rings have on the molecule's property, which is the primary issue with our current GAT model.

8 Conclusions

In this report, we investigated the problem of molecular property prediction. We built an attention-based GAT model for predicting the Tox21 dataset, and showed that it outperformed previous baseline models. Subsequently, we analyzed the results of the model, and found that it performed well on small, linear molecules, but struggled on larger, aromatic ones. Our GAT model represents a new stride in molecular property prediction by incorporating attention over neighboring

nodes. In the future, we believe that the model can be improved even further by introducing features for aromaticity and other relevant attributes.

9 Contributions

Both team members contributed equally to the project.

10 Code

Publicly accessible: https://github.com/rohanm2/CS-230-Final-Project-

References

[1] Duggan, Susan, and Norma Pence. "A Tough Road: Cost To Develop One New Drug Is $2.6 Billion; Approval Rate for Drugs Entering Clinical Development Is Less Than 12

[2] National Toxicology Program. Annual Report for Fiscal Year 2016. Research Triangle Park, NC: National Toxicology Program; 2017. Available from https://ntp.niehs.nih.gov/annualreport/2016

[3] Mayr, Andreas, et al. "DeepTox: Toxicity Prediction Using Deep Learning." Frontiers in Environmental Science, vol. 3, 2016, doi:10.3389/fenvs.2015.00080.

[4] Huang, R. (2019). Predictive Modeling of Tox21 Data. In Challenges and Advances in Computational Chemistry and Physics (pp. 279–297). Springer International Publishing. https://doi.org/10.1007/978-3-030-16443-0-14

[5] Thomas Unterthiner, Andreas Mayr, Günter Klambauer: "Toxicity Prediction using Deep Learning", 2015; [http://arxiv.org/abs/1503.01445 arXiv:1503.01445].

[6] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing: "MoleculeNet: A Benchmark for Molecular Machine Learning", 2017; [http://arxiv.org/abs/1703.00564 arXiv:1703.00564].

[7] Michaël Defferrard, Xavier Bresson: "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering", 2016, Advances in Neural Information Processing Systems 29 (2016); [http://arxiv.org/abs/1606.09375 arXiv:1606.09375].

[8] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò: "Graph Attention Networks", 2017; [http://arxiv.org/abs/1710.10903 arXiv:1710.10903].

[9] Minkin, V. I., Mikhail N. Glukhovtsev, and B I. Simkin. Aromaticity and antiaromaticity : electronic and structural aspects. New York: Wiley, 1994. Print.