Improving QSAR Modeling for Predictive Toxicology using Publicly Aggregated Semantic Graph Data and Graph Neural Networks

Joseph D. Romano*, Yun Hao*, and Jason H. Moore†

Institute for Biomedical Informatics, University of Pennsylvania, Philadelphia, Pennsylvania 19104, United States

†*E-mail: jhmoore@upenn.edu*

**These authors contributed equally.*

Quantitative Structure-Activity Relationship (QSAR) modeling is the most common computational technique for predicting toxicity for specific chemicals, but a lack of methodological innovations have led to underwhelming performance of QSAR in modern applications. We show that contemporary QSAR modeling for predictive toxicology can be substantially improved by incorporating semantic graph data aggregated from open-access public databases, and analyzing those data in the context of graph neural networks (GNNs). Furthermore, we introspect the GNNs to demonstrate how they can lead to more interpretable applications of QSAR, and use ablation analysis to explore the contribution of different data elements to the final models' performance.

Keywords: Toxicology; Graph neural networks; Data aggregation; QSAR; Artificial intelligence.

1. Introduction

Evaluating the toxicity of chemicals is an essential component of pharmaceutical and environmental research. Traditionally, the task of establishing toxicity has been accomplished using *in vivo* models, where a model organism is exposed to a chemical of interest and observed for toxic effects, or by performing retrospective observational studies on human populations in the context of epidemiological analyses. Both of these approaches are costly and time consuming, and given the hundreds of thousands of compounds of toxicological interest, innovative alternatives are needed to rapidly screen chemicals. In recent decades, predictive toxicology and large-scale chemical screening efforts have emerged to address this issue.

Quantitative Structure-Activity Relationship (QSAR) modeling is arguably the most common method for predicting *in silico* whether a chemical will cause a toxic response. Briefly, QSAR modeling involves collecting quantitative descriptions of molecular structures, and then fitting a statistical model (e.g., logistic regression, or other supervised machine learning algorithms) to chemicals where a toxic endpoint of interest is already known. Since each data point used to train a model is itself the outcome of a single experiment, QSAR is a meta-analysis approach that is complicated not only by the challenge of capturing relevant structural features of chemicals, but also by errors, biases, and ambiguities in the underlying experiments used to generate the training data. Consequently, QSAR is heavily criticized for its disappointing performance on many tasks. The computational toxicology community has long acknowledged the need for new methodological innovations to improve QSAR performance, but few have emerged in the past several decades.

In this study, we address these issues by augmenting the traditional QSAR approach with multimodal graph data aggregated from several public data sources, and analyzing those data in the context of a heterogeneous graph convolutional neural network (GCNN) model. We evaluate the model using 59 assays and their accompanying chemical screening data from the Tox21 data repository, and compare its performance to two rigorously defined traditional QSAR models consisting of random forest and gradient boosting classifiers. Our results show that the GNN strategy significantly outperforms traditional QSAR. We further refine our results by using an ablation analysis to explain the relative contributions of different data sources to the GNNs' increased performance. Finally, we discuss how GNNs improve the interpretability of QSAR, and suggest future directions.

2. Methods

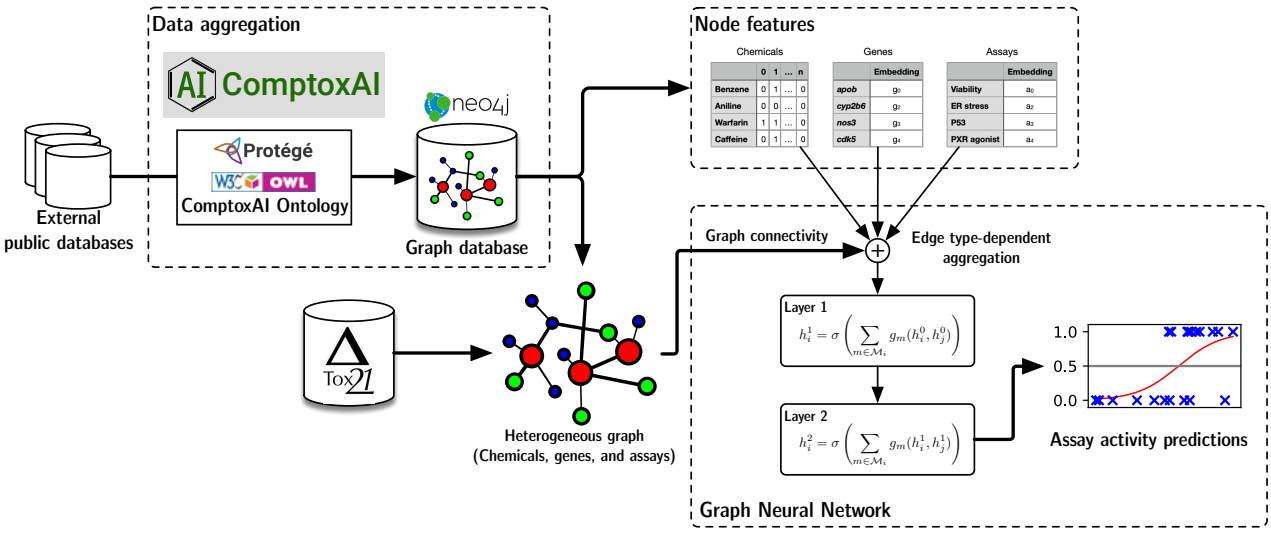

Fig. 1. Schematic overview of the graph machine learning approach used in this study. We build a toxicology-focused graph database (named ComptoxAI) using data aggregated from many public databases, and extract a subgraph to be used for QSAR analysis (containing chemicals, assays, and genes). We then train and evaluate a graph convolutional neural network on the heterogeneous graph for predicting whether or not a chemical activates specific toxicology-focused assays from the Tox21 database.

2.1. *Aggregating publicly accessible multimodal graph data*

The graph data used in this study come from a new data resource for computational toxicology, named ComptoxAI. ComptoxAI includes a large graph database containing many entity and relationship types that pertain to translational mechanisms of toxicity, all of which are sourced from third-party public databases (including PubChem, Drugbank, the US EPA's Computational Toxicology Dashboard, NCBI Gene, and many others). We extracted the subgraph from ComptoxAI's graph database defined as all nodes representing chemicals, genes,

and toxicological assays, as well as the complete set of edges linking nodes of those types. A metagraph describing the node and edge types in the subgraph is shown in [FIGURE XXX].

Every prediction task in this study involves QSAR modeling between chemicals of toxicological interest and toxicology-related activity assays derived from the US EPA's Tox21 dataset. Each chemical and assay is represented as a node within the larger graph database, and all chemical–assay pairs are either linked by an edge (*chemicalHasActiveAssay* or *chemicalHasInactiveAssay*) or not linked by an edge (i.e., no data has been collected on the chemical with respect to the assay in question).

2.2. *Obtaining toxicology assay data*

We used the Tox21 dataset—which is a freely available resource produced collaboratively by the US National Institutes of Health, the US Food and Drug Administration, and the US Environmental Protection Agency—to obtain a set of candidate assays for classification and establish 'ground truth' relationships between specific chemicals and those assays.

2.3. *Heterogeneous graph neural network*

We constructed a heterogeneous graph convolutional neural network (GCNN) architecture for the graph ML experiments. Since our approach uses multiple entity types (e.g., chemicals, genes, assays) in the same graph—each with possibly different sets of node features, and linked by multiple semantically distinct edge types—the architecture extends the common GCNN model to learn separate message passing functions for each edge type. Briefly, each layer of the network aggregates signals from adjacent nodes in the graph, such that a greater number of layers results in signals being aggregated from an increasingly wider radius around each node. The output of the network can be thought of as encoded representations of nodes that incorporate information from other nodes in their local neighborhood. Additionally, GCNNs can be thought of as a generalization of convolutional neural networks (CNNs) used in computer vision—instead of the convolutional operator aggregating signals from nearby pixels in an image, it aggregates features from adjacent nodes in the graph.

In a heterogeneous graph, different node types represent different types of entities, each represented within a semantically distinct feature space. Therefore, the process of aggregating information from adjacent nodes must take those nodes' types into account. Additionally, different edge types (e.g., 'chemicalUpregulatesGene' and 'chemicalDownregulatesGene') convey their own semantically distinct meanings, which can substantially effect the flow of information through the network. To handle these two challenges, we learn separate aggregation functions for each edge type in the graph, following the example proposed by Schlichtkrull *et al* in R-GCNs (Relational Graph Convolutional Networks).[1] Within the R-GCN paradigm, the aggregation process can be split into 3 sequential steps: (1.) collecting signals from adjacent nodes using an appropriate edge type-specific message function ϕ, (2.) combining each of those incoming signals (across all edge types) via a reduce function ρ, and (3.) finally updating the target node v by applying an update function ψ. Training the network is roughly equivalent to finding an appropriate parameterization of ϕ for each edge type.

A formal description of the GNN is given in **Appendix A**.

2.3.1. *Node classification model*

Our node classification task consists of labeling chemicals according to whether they do (1) or do not (0) activate a specific Tox21 assay. The procedure we use for generating these labels is as follows:

(1) For each $c \in \mathcal{C}$ is adjacent to the assay of interest $a \in \mathcal{A}$, generate labels according to the following scheme:

   (a) 1 if there exists an edge (c, r, a) such that the edge type of r is `chemicalHasActiveAssay`.
   (b) 0 if there exists an edge (c, r, a) such that the edge type of r is `chemicalHasInactiveAssay`.
   (c) If both of the two preceding conditions are false, no label is generated.

(2) Delete a (and all edges incident to a) from the graph.

The resulting graph $\mathcal{G}_a^\star$ is then used in a standard supervised learning task where the goal is to predict the labels on nodes corresponding to chemicals, which represent whether the assay is or is not activated in response to perturbation by that chemical. We use an 80%/20% train/test split on the labeled chemicals, optimize the GCNN's parameters using the Adam algorithm (a computationally efficient variant of stochastic gradient descent suitable for sparse gradients),[2] and compute the error between predicted and true labels via cross entropy loss.

To assess the contribution of the MACCS molecular descriptors when added to the GNN, we trained the node classification model both in the presence and in the absence of MACCS bitstrings applied as node features to each chemical. Intuitively, the model trained without MACCS node features performs inference using only the graph's topological structure—including gene interactions and activity annotations to the other Tox21 assays—while the one with MACCS node features additionally has access to the same chemical structure information used in the non-graph (baseline) QSAR models.

Specific details for the node classification task are given in **Appendix B**.

2.4. *Baseline QSAR classifiers*

To assess the relative performance of the GNN classification model, we built 2 additional (non-NN) QSAR models that represent well-established current practice in predictive toxicology: A random forest classifier, and a gradient boosting classifier. Each model was trained on the aforementioned MACCS fingerprints of chemicals computed from SMILES strings, with an 80%/20% training/testing split. We tuned 6 hyperparameters for each random forest model, and 5 for each gradient boosting model, as described in **Table S1**. These were tuned using grid search, where the optimal hyperparameter set is defined as the one minimizing binary cross entropy between predicted labels and true labels on the training data.

3. Results

3.1. *GNN node classification performance vs. baseline QSAR models*

Of the 68 total assays in the Tox21 database, we retained 52 for analysis in the QSAR experiments. The remaining 16 assays were not used due to either a low number of active chemicals or underrepresentation of screened chemicals in the ComptoxAI graph database. Furthermore, we discarded compounds with inconclusive or ambiguous screening results.

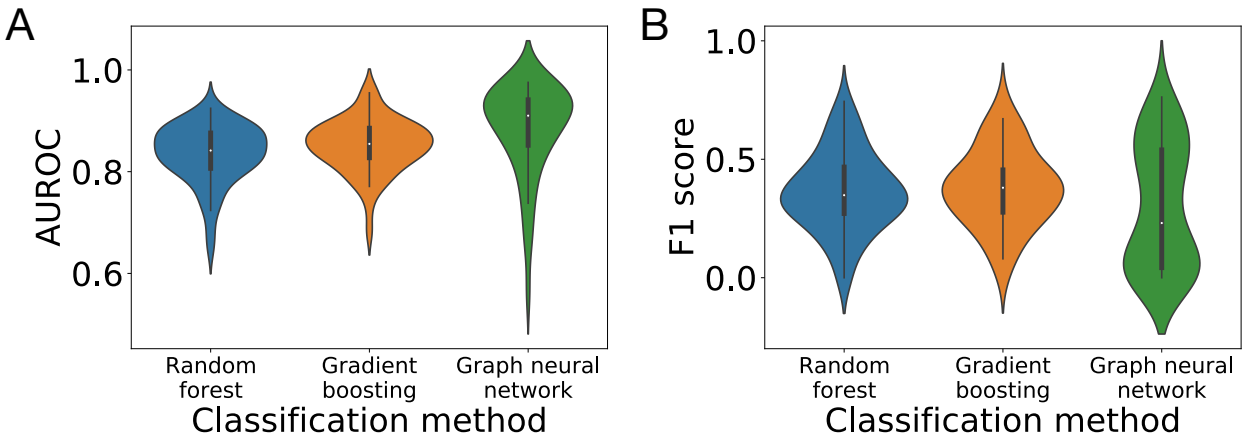

Fig. 2. Overall performance metrics of the 3 QSAR model types on each of the Tox21 assays—a.) Area Under the Receiver Operator Characteristic curve (AUROC) and b.) F1 score. The mean AUROC is significantly higher for the GNN model than for either of the two baseline QSAR approaches. The differences in F1 scores are not statistically significant. It is worth noting that the GNN achieves poor F1 scores on assays with relatively few (e.g., < 100) "active" annotations in Tox21, which is consistent with known performance of neural networks on data with sparse labels.

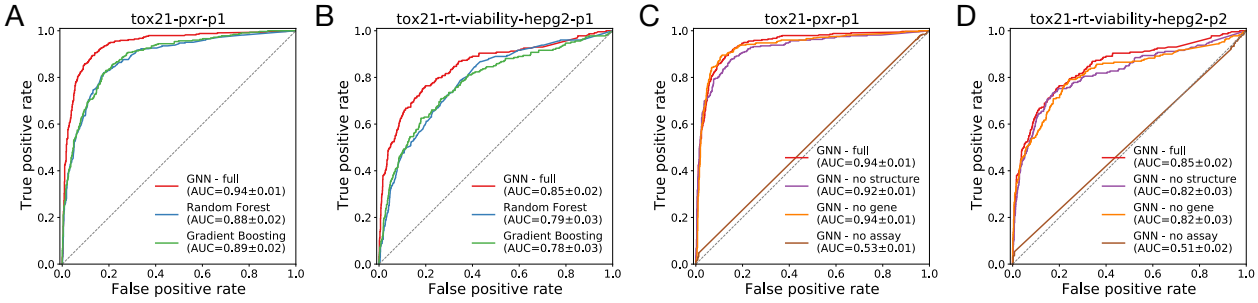

Fig. 3. Receiver Operator Characteristic (ROC) curves for two selected Tox21 assays: a.) PXR agonism (`tox21-pxr-p1`) and b.) HepG2 cell viability (`tox21-rt-viability-hepg2-p1`). In both cases, the area under the curve (AUC) is significantly higher for the GNN model than either the Random Forest or Gradient Boosting models. Cell viability assays—in particular—are notoriously challenging to predict computationally.

3.2. *Ablation analysis of graph components' influence on the trained predictive model*

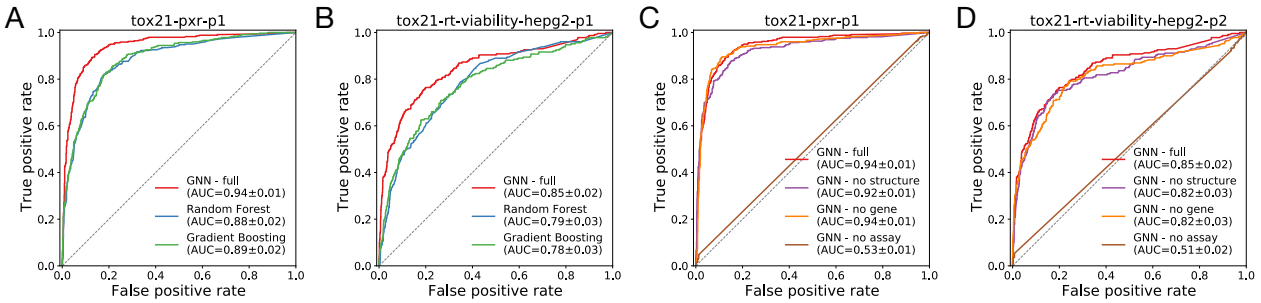

Fig. 4. Receiver Operator Characteristic (ROC) curves for two selected Tox21 assays using different configurations of the GNN model. 'GNN - full' is the complete model as described in §2.3.1. 'GNN - no structure' omits the MACCS chemical descriptors and replaces them with node embeddings of the same dimensionality. 'GNN - no gene' removes gene nodes and their incident edges from the network. 'GNN - no assay' removes all assay nodes and incident edges, so predictions are made solely using chemicals, genes, the remaining edges, and the MACCS chemical descriptors as chemical node features. For both assays, the full model performs the best, while the model without assay nodes performs only marginally better than guessing labels at random.

3.3. *Interpretability of trained GCNNs via assay embeddings*

4. Discussion

5. Conclusions

6. Code availability

All source code pertaining to this study is available on GitHub at `https://github.com/EpistasisLab/qsar-gnn`. A frozen copy of the code at the time of writing is available at [XXX; Zenodo].

7. Supplemental Materials

Supplemental tables and figures are available on FigShare at [XXX].

Acknowledgements

This work was made possible with support from US National Institutes of Health grants `R01-LM010098`, `R01-LM012601`, `R01-AI116794`, `UL1-TR001878`, `UC4-DK112217` (PI: Jason Moore), `T32-ES019851`, and `P30-ES013508` (PI: Trevor Penning).

References

1. M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov and M. Welling, Modeling relational data with graph convolutional networks, 593 (2018).

2. D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980* (2014).

Appendix A. Graph convolutional network architecture

Some stuff here.

Appendix B. Node classification model

Some more stuff.