Reaction Prediction Using Graph Convolutional Networks

ENSC 813 Course Project

Brett Hannigan
bchannig@sfu.ca

2019-04-08

1 Introduction

1.1 Problem Definition

Reaction prediction and its inverse problem, retrosynthesis, are important in the field of synthetic chemistry. Since the 1970s, algorithmic approaches to solving these problems have been developed, primarily using expert systems [1]. The high dimensionality of the problem may lend itself well to deep learning as a way to avoid the time consuming task of developing rule-based systems while also delivering higher performance than simulation-based methods.

Deep learning has been applied to chemistry in many ways. For drug discovery, an autoencoder of chemical structures can be used to map a compound into a latent space. This space can be used for exposing similar molecules with more optimal properties for a specific goal [2]. Graph convolutional networks have been used to generate molecular "fingerprints" [3] — condensed representations of a molecule's structure. Neural network models that predict certain features, such as reactivity, toxicity, or parameters of pharmacological interest, have also been researched with considerable success [4].

1.2 Related Work

Applied to reaction prediction, Wei et al. [5] used a multilayer perceptron classifier to predict the reaction type given chemical fingerprint vectors of each reactant. The products were then determined using a look-up table of standard reaction transformations. The dataset was synthesized and resembles simple textbook questions. The group did not claim a definitive accuracy, rather measured the similarity score between the predicted and actual products. Liu et al. [6] used a "sequence-to-sequence" recurrent neural network to map reactants to products described using SMILES (simplified molecular-input line-entry system) strings. Trained on 50 000 of the 1.8 million reactions scraped from patent data [7], the model achieved a top-1 accuracy of 37.4 percent. In 2018, Jin [8] published a model using a Weisfeiler-Lehman difference network that correctly predicted nearly 80 percent of the validation reactions from the same patent dataset. In this case, molecules were represented as graphs. Finally, Schwaller et al. [9] developed another recurrent model operating on SMILES-encoded chemical data that achieved a slightly higher accuracy of 80.3 percent. These similar works are listed in Table 1.

1.3 Data Representation

As seen in the related work, chemical reaction data can be represented in a variety of ways. SMILES is ubiquitous as it is compact and human-readable, but it is very sensitive to formatting. In the case of machine learning, the model must learn both the atom mapping from reactants to products as well as the SMILES grammar. The related work that used SMILES strings claims between 1.3% and 12.2% of output SMILES were syntactically invalid. In contrast, 2-D chemical structures are easily represented as graphs, which would be less susceptible to generating erroneous output. A non-directed graph for representing compounds with n atoms may be defined as $\mathcal{G}(A, X)$, where $A^{n \times n}$ is the ad-

Table 1: A list of some recent work on reaction prediction using machine learning.

Ref.	Year	Input	Model	Dataset	Top-1 Acc.
[5]	2016	Fingerprint vector	Multilayer Perceptron	Synthesized	N/A
[6]	2017	SMILES string	Seq2seq RNN	USPTO-50K	37.4%
[8]	2018	Graph	Weisfeiler-Lehman network	USPTO [7]	79.6%
[9]	2018	SMILES string	Long short-term memory RNN	USPTO [7]	80.3%

jacency matrix and $X^{n\times m}$ is the feature matrix consisting of m features per atom. When there are multiple molecules involved, A has block diagonal structure.

1.4 Graph Convolution

Graph convolution is often done spectrally by taking a Fourier transform of the graph so that convolution may be done via multiplication in the frequency domain. Due to the high computational complexity of this method, there is considerable research into approximating convolution over graphs. A common approach uses a Chebyshev polynomial approximation that reduces computation time [10]. A modification of this method known as fast approximate convolution has the propagation function of the form [11]:

$$H^{(0)} = X \tag{1}$$

$$H^{(l+1)} = \sigma\left(D^{-\frac{1}{2}}\left(A + I_n\right)D^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{2}$$

where $H^{(l)}$ is the output of layer l with associated weight matrix $W^{(l)}$, σ is the activation function, D is the degree matrix of the graph, and the identity matrix is added to A to introduce self-loops, otherwise each node's own features are not propagated to itself. The diagonal nature of D makes the square-root operation trivial and the entire propagation function computationally efficient.

2 Methods

2.1 Data Preprocessing

The Lowe patent dataset consisting of reaction SMILES was parsed using RDKit [12]. Of the 1 808 937 reactions, 2656 were skipped because they either had greater than 128 atoms on the reactant or product sides or had inconsistencies when interpreted by RDKit. For the feature matrix, the features listed in Table 2 [3, Table 2] were selected and extracted using RDKit. A Python script was used to convert the RDKit molecule objects into graphs using the tool NetworkX [13]. From this, sparse A and X matrices were saved for both the reactant and product sides of each reaction to be used in training and validation with an 80%-20% split. Due to training time constraints, the training and validation sets had to be reduced by a factor of 250, resulting in 5780 training and 1445 test examples, randomly selected.

2.2 Network Architecture

A graph convolutional network (GCN) was built in Keras using a custom layer derived from an implementation found online[1]. The network consists of two graph convolutional layers followed by a pair of parallel 3-layer multilayer perceptrons. The first of these operates on the output of the convolutional block and generates the output feature matrix. The second operates on the concatenation of the convolutional output with the input adjacency matrix and generates the output adjacency matrix. The topology of the network is shown in Figure 1. The tensor sizes at each layer are given in Table 3.

Table 2: List of extracted features as columns in feature matrix X.

Feature Name	Encoding	Size
`atomic_num`	One-hot over H, C, N, O, F, P, S, Cl, Br, or null (other)	9
`chiral_tag`	One-hot over R- or S- chirality, or null (achiral)	2
`hybridization`	One-hot over s, sp, sp^2, sp^3, or null (other)	4
`formal_charge`	Integer	1
`is_aromatic`	1 if in aromatic ring, 0 otherwise	1
`num_explicit_h`	Integer	1

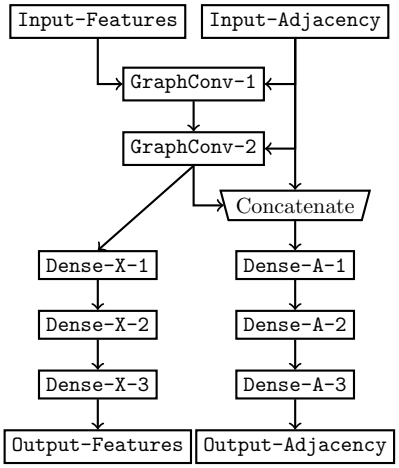


Figure 1: Diagram of the layer topology of the graph convolutional network.

Table 3: The tensor sizes at each layer of the network with a batch size of 1024 and a maximum of 128 graph nodes.

Layer	Dimensions
`Input-Features`	(1024, 128, 18)
`Input-Adjacency`	(1024, 128, 128)
`GraphConv-1`	(1024, 128, 1024)
`GraphConv-2`	(1024, 128, 1024)
`Dense-X-1`	(1024, 128, 512)
`Dense-A-1`	(1024, 128, 512)
`Dense-X-2`	(1024, 128, 256)
`Dense-A-2`	(1024, 128, 256)
`Dense-X-3`	(1024, 128, 18)
`Dense-A-3`	(1024, 128, 128)
Total Parameters	2 484 370

Table 4: Model hyperparameters.

Activation	`GraphConv`	ReLU
	`Dense-X-1` `Dense-A-1`	tanh
	`Dense-X-2` `Dense-A-2`	tanh
	`Dense-X-3` `Dense-A-3`	linear
Loss	Mean-squared error	
Optimizer	Adam	
Learning rate	0.001	
Batch size	1024	
Early stopping	Patience	10

3 Results

3.1 Partition Coefficient Regression

3.1.1 Problem Statement

Most research involving graph convolutional networks apply them to single, large graphs such as citation networks or social networks. To evaluate whether this type of model is well suited for cheminformatics tasks composed of many smaller graphs, a simpler problem was first considered. The Kaggle LogP dataset [14] contains 14 609 records of chemical structures in SMILES format and their respective water-octanol partition coefficient, a measure of hydrophobicity important in pharmacology. Being a more straightforward regression problem, a model for predicting partition coefficients from chemical structure was first implemented to test feasibility of the approach.

[1] https://github.com/CyberZHG/keras-gcn

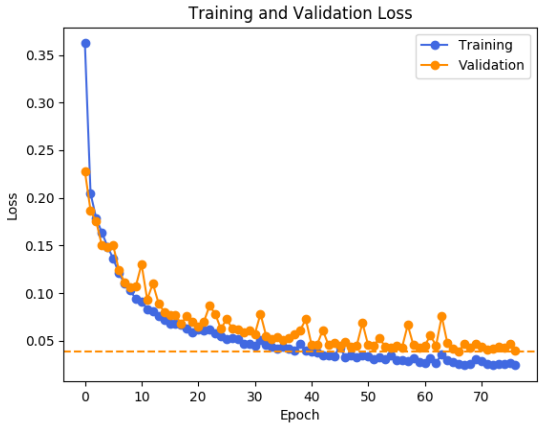


Figure 2: The training and validation loss for the partition coefficient regression problem.

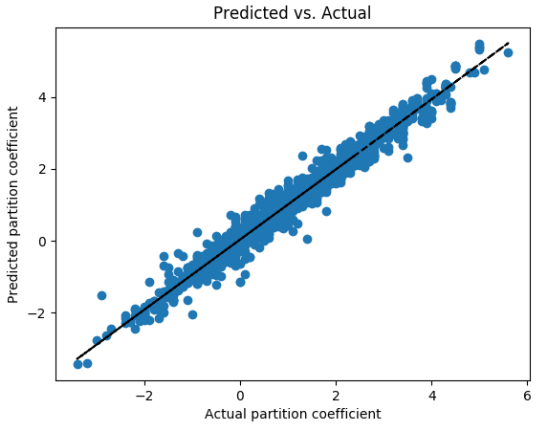


Figure 3: The predicted partition coefficients versus true value for the validation set and the best linear fit.

3.1.2 Model Architecture and Training

This model was covered in the project presentation, so just the main design and results are described here. The network architecture was similar to that of Figure 1 except using one dense branch and having a global average pooling layer at the end to provide a single output value. The network was trained using an 80%-20% training-validation split with a batch size of 32. Each epoch took approximately 90 s when training on the CPU (8×2.8 GHz Intel Xeon, 12 GB RAM) and the training was terminated when the early stopping criterion was met at the 77^{th} epoch as shown in Figure 2.

3.1.3 Partition Coefficient Results

The results on the validation set are shown in Figure 3, indicating a root-mean-squared error (RMSE) of 0.04. A Bland-Altman plot comparing the predicted and actual partition coefficients is shown in Figure 4 where the 95% confidence interval is about 0.4. The results on this dataset surpass those reported in literature using classical machine learning methods, the best of which have an RMSE of 0.86 [15]. This is evidence that the graph convolutional network is able to generalize the features of a compound and thus the reaction prediction problem was pursued.

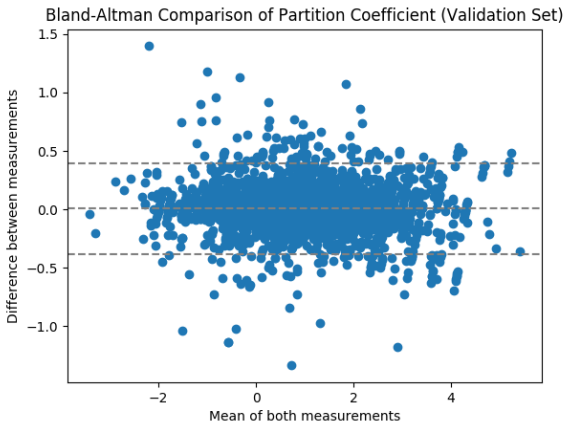


Figure 4: A Bland-Altman plot comparing the predicted partition coefficient using the GCN model with the true value.

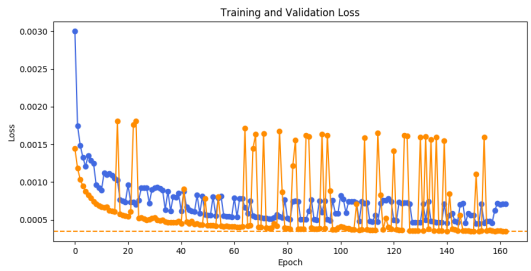


Figure 5: The training and validation loss for the reaction prediction problem.

3.2 Reaction Prediction

3.2.1 Training

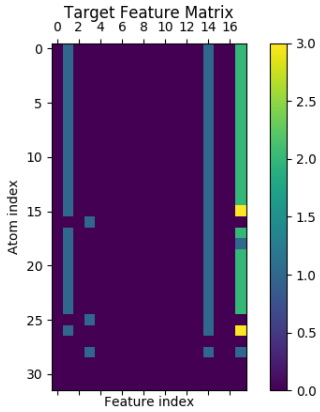
Having established that the GCN model may be used on chemical structure data with considerable success, the original problem of reaction prediction is resumed. The model was trained on the subset of data from Section 2.1. The training was terminated when 10 epochs elapsed without improvement in validation loss. This occurred at the 163$^{\text{rd}}$ iteration, each of which took approximately 425 s. The plot of training and validation losses is shown in Figure 5. With no significant overfitting being evident, the addition of dropout or regularization was not explored. The average mean-squared error (MSE) between the validation adjacency matrices was 5.6×10^{-5} and between the validation feature matrices was 2.9×10^{-3}.

3.2.2 Case Studies

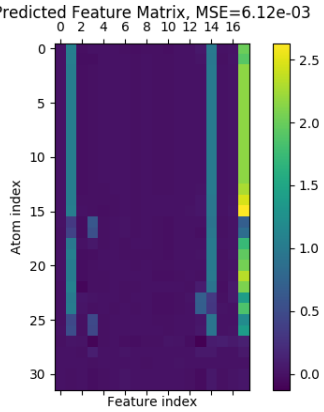
As described in Section 2.2, the input to the network is an adjacency and feature matrix of the reactants and the desired output is the adjacency and feature matrix of the products. In this model, features are at the node-level rather than edge-level, meaning that the order of each bond (e.g. single, double, triple) is not explicitly featurized. Despite this, the presence of the `hybridization` and `formal_charge` atom features make assigning bond orders a relatively straightforward chemistry exercise using valence bond theory. Due to time constraints, a robust algorithm for this was not implemented, and therefore it a top-1 accuracy cannot be specified. Instead, two example data are presented, chosen from a random sample of 10 validation records. By comparing the MSE of these examples to the validation global average, a rough idea of the accuracy may be obtained.

The first example has a target product feature matrix presented in Figure 6a. Run on this datum, the model returns the feature matrix presented in Figure 6b. The prediction mean-squared error (MSE) of 6.12×10^{-3} is about twice that of the global validation average for this example. The errors are primarily in the `num_explicit_hs` feature. The target and predicted adjacency matrices are shown in Figures 7a and 7b, respectively, where an MSE of 4.60×10^{-5} was achieved, close to the global validation average. The prediction is very accurate except for the last 5 atoms which are not present in the product. The Lowe patent dataset omits side products and as such the reactions aren't balanced. In this case, the model does not show zeros place of these missing atoms in the adjacency matrix.

The second example has target and predicted feature matrices as seen in Figures 8a and 8b. In this case, the feature matrices are visually very close and have an MSE of 2.37×10^{-3}, again better than the global validation average. In the case of the adjacency matrices seen in Figures 9a and 9b, a similar result to the previous example is seen. It is of note that there is a bug in the data preprocessing code, as there appears blank regions in some input adjacency matrices, such as that of Figure 9a. There was not enough time to correct this error and also re-train the model, although this would also explain some of the discrepancies of the diagonal elements between the target and predicted adjacency matrices. In practice, the accuracy is likely to be higher because a post-processing step can trim orphan atoms (those with edges only on the adjacency matrix diagonal) and use other heuristics to complete the most probable structure from the feature and adjacency data.

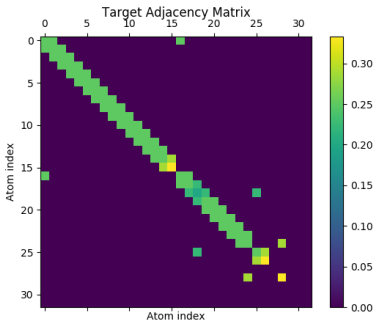


(a) The target product feature matrix.

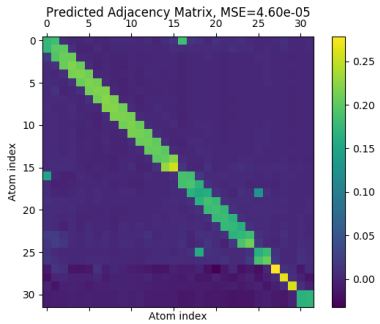


(b) The predicted product feature matrix.

Figure 6: Comparison of predicted and target feature matrices for validation data ID: 272 276.

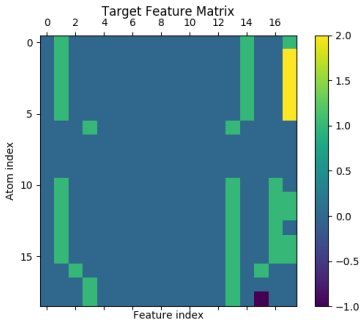


(a) The target product adjacency matrix.

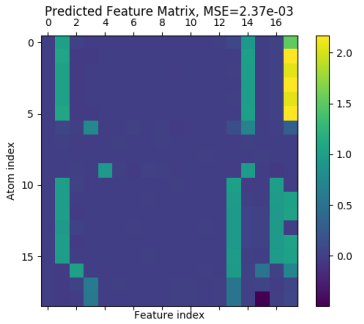


(b) The predicted product adjacency matrix.

Figure 7: Comparison of predicted and target adjacency matrices for validation data ID: 272 276.

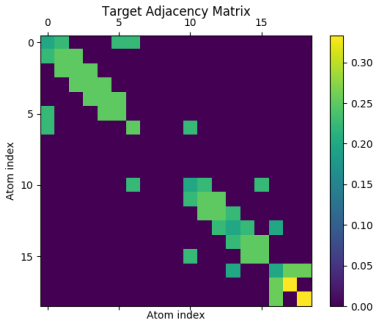


(a) The target product feature matrix.

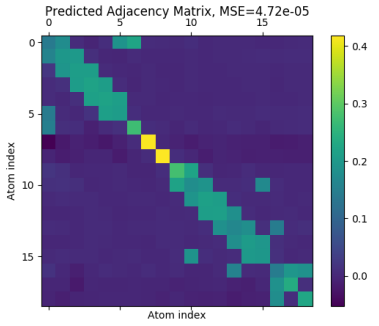


(b) The predicted product feature matrix.

Figure 8: Comparison of predicted and target feature matrices for validation data ID: 1 203 755.



(a) The target product adjacency matrix.



(b) The predicted product adjacency matrix.

Figure 9: Comparison of predicted and target adjacency matrices for validation data ID: 1 203 755.

4 Conclusion

This project achieved the following:

- Applied deep learning to cheminformatics data.
- Used graph convolutional networks to perform regression and prediction tasks.
- Provided a proof-of-concept that the architecture is capable of predicting the outcome of chemical reactions as trained on a small subset of data.

4.1 Future Work

The large amount of computation time required for training limited the scope of the project. The most glaring improvements needed are to correct the data parsing error and to implement a method to convert an adjacency and feature matrix back to a molecule object to better compare predictions. If computational resources permit, it would be very interesting to train on the entire dataset and observe the change in loss functions. During training, the use of dropout and regularization may be explored, especially ℓ_1 regularization to promote sparsity in the adjacency matrices. The adjacency matrix may also be normalized so that its columns sum to one allowing the use of the cross-entropy loss function. Finally, for performance improvements, the GCN can be optimized to use sparse matrix data structures and to store the preprocessed data uncompressed for faster calculation and loading.

A Script Listing

- `preprocess.py` – Reads Lowe patent data into an RDKit Mol object, sanitizes the Mol object, produces a `ReactionSideGraph` object for each reaction, and writes the adjacency and feature matrices of the products and reactants to file.
- `ReactionGraph.py` – Custom class that stores one side of a chemical reaction as a NetworkX object, provides logic used by `preprocess.py` to produce adjacency and feature matrices.
- `convert_rdkit_to_networkx.py` – Function to extract features of interest from an RDKit Mol object and produce a NetworkX graph, used by the `ReactionSideGraph` constructor.
- `ReactionData.py` – Custom `Keras.Sequence` object to read in data.
- `test_gcn.py` – Reads a small subset of 8 randomly selected validation data, predicts the output using the pre-trained model, and optionally produces the feature and adjacency matrix graphs seen in Figures 6 to 9.
- `utils.py` – Contains two short functions for graphing and generating figures.
- `20190408T181100_ReactionPrediction.h5` – The pre-trained model with saved pa-

rameters, requires the `GraphConv` custom layer.

- `/keras-gcn/` – The class implementing the Keras GCN layer.

- `/test_gcn_data/` – A set of 10 randomly selected validation data for `test_gcn.py`, split into sub-folders `reac` for reactants and `prod` for products, each of which contains another sub-folder `feat` for the corresponding feature matrix and `adj` for the adjacency matrix, all stored in sparse `.npz` format.

References

[1] E. J. Corey and W. Todd Wipke, "Computer-assisted design of complex organic syntheses," *Science*, vol. 166, no. 3902, pp. 178–192, 1969.

[2] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik, "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules," *ACS Central Science*, vol. 4, no. 2, pp. 268–276, 2018.

[3] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular graph convolutions: moving beyond fingerprints," *Journal of Computer-Aided Molecular Design*, vol. 30, no. 8, pp. 595–608, 2016.

[4] C. W. Coley, W. Jin, L. Rogers, T. F. Jamison, W. H. Green, T. Jaakkola, R. Barzilay, and K. F. Jensen, "A graph-convolutional neural network model for the prediction of chemical reactivity," *Chemical Science*, 2019.

[5] J. N. Wei, D. Duvenaud, and A. Aspuru-Guzik, "Neural networks for the prediction of organic chemistry reactions," *ACS central science*, vol. 2, no. 10, pp. 725–732, 2016.

[6] B. Liu, B. Ramsundar, P. Kawthekar, J. Shi, J. Gomes, Q. Luu Nguyen, S. Ho, J. Sloane, P. Wender, and V. Pande, "Retrosynthetic reaction prediction using neural sequence-to-sequence models," *ACS central science*, vol. 3, no. 10, pp. 1103–1113, 2017.

[7] D. Lowe, "Chemical reactions from US patents (1976-Sep2016)," 2017.

[8] Wengong Jin, "Neural Graph Representation Learning with Application to Chemistry," *Master Thesis*, 2018.

[9] P. Schwaller, T. Gaudin, D. Lányi, C. Bekas, and T. Laino, ""Found in Translation": predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models," *Chemical Science*, vol. 9, no. 28, pp. 6091–6098, 2018.

[10] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun, "Graph Neural Networks: A Review of Methods and Applications," pp. 1–20, 2018.

[11] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," pp. 1–14, 2016.

[12] G. Landrum, "RDKit: Open-source cheminformatics."

[13] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network ntructure, nynamics, and nunction using NetworkX," *Proceedings of the 7th Python in Science Conference*, no. SciPy, pp. 11–15, 2008.

[14] MatthewMasters, "logP of Chemical Structures — Kaggle," 2018.

[15] E. W. Lowe, M. Butkiewicz, M. Spellings, A. Omlor, and J. Meiler, "Comparative analysis of machine learning techniques for the prediction of logP," *IEEE SSCI 2011 - Symposium Series on Computational Intelligence - CIBCB 2011: 2011 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, no. 3, pp. 35–40, 2011.