Ethan Curtis

Final Project Report

https://github.com/ethancurtis/Basis-Set-Superposition-Error-cs230/

Problem Statement

In quantum chemistry, molecular wavefunctions are represented as a linear combination of hydrogen-like wavefunctions (orbitals) centered on each atom. The reduction in dimensionality from the formally infinite dimensional Hilbert space of the molecular wavefunction to a finite basis set introduces some error into the energy of the wavefunction, known as basis set incompleteness error (BSIE). Reaction modeling often deals with multiple molecules in close proximity. If one were to acquire the perfect BSIE correction for energies of single molecules and apply it to a real chemical system of several molecules, one would find that the correction is no longer perfectly accurate! This effect is known as basis set superposition error (BSSE). Electrons on one molecule are borrowing unoccupied orbitals on other molecules to effectively expand their basis set beyond the basis set for the isolated molecule (and lower the energy). This effect is unphysical because increasing the basis set size localizes these electrons back onto their respective molecules. The result of BSSE is to overestimate the strength of interaction of two nearby molecules (or even nearby parts of one large molecule), which can be problematic for studying chemical reactions. Designing a BSSE correction for a small basis set would allow for simulations to use that small basis set (allowing them to run more quickly) and still obtain accurate results.

Background

Boys and Bernardi defined a basis set superposition error (BSSE) correction for pairs of molecules (Boys and Bernardi Counterpoise, or BBCP).[1] For each molecule in the system, they calculated the energy of the isolated molecule with that molecule's basis set, and with the basis set for the entire system. The difference between these two quantities is the BSSE (the stabilization derived from accessing another molecule's basis set). Jensen extended this work for BSSE involving any two atoms, not just two separated molecules.[2] However, these calculations are often too time-consuming to be practical because they require many wavefunction calculations. In 2012, Grimme and Kruse developed a fast empirical method for estimating BSSE based on parameterization of a simple atom-pairwise orbital overlap model.[3] They named it geometric counterpoise (gCP) because it only depends on the geometry of the molecule and a set of parameters. Grimme and Kruse fit their empirical correction to the Boys and Bernardi correction for a set of 66 dimers (with 8 geometries per dimer for a total of 528 data points). This model has four parameters per choice of method and basis set. These parameters were fit to a dataset containing 528 data points. The small size of the parameter set and training set for the state-of-the-art suggests that there is much to be gained from a large dataset machine learning approach.

Dataset

One of the key contributions of this project is the construction of a dataset of non-covalently bound small molecule dimers. Simplified Molecular Input Line Entry System (SMILES) strings encode chemical structures in text strings.[4] The GDB-11[5] and GDB-13[6] datasets provide lists of SMILES strings for

small, drug-like molecules (containing elements H, C, N, O, F, S, and Cl). All the molecules containing 11 or less heavy elements (not H) were concatenated into a list of 42,569,450 SMILES strings. To convert the SMILES strings into XYZ coordinates of molecules, the rdkit Python package was employed. From this text file, two strings are drawn at random and converted to XYZ coordinates (due to the expected size of my dataset, I am not concerned about the chance of choosing the same two molecules twice). The molecules are fed into Packmol,[7] a program which takes in the two sets of XYZ coordinates and produces a single set of XYZ coordinates containing the dimer (with the molecules randomly oriented). The dimer geometry is optimized by Amber,[8] a molecular mechanics package. The data set is augmented by repeating the coordinates → Packmol → Amber process 4 times, producing 4 unique dimer geometries for each pair of molecules. Finally, the coordinates are ready for labeling.

For this project, BBCP is calculated with Hartree-Fock theory (HF) and the MINIS[9] basis set. BSSE depends on the choice of theory and basis set, so any correction (empirical, NN, etc.) will need to be re-trained for each choice of theory. HF/MINIS was chosen as the method to be studied for this project because it seems to be a promising compromise between speed and accuracy, provided that its substantial BSSE can be corrected,[3] and gCP parameters for this method already existed. TeraChem[10, 11] was used to calculate the energies of the dimers to determine the BSSE. About 1% of the structures fail to be chemically reasonable, and these are removed by comparing their energies before and after the Amber geometry optimization (extremely large energy changes during optimization indicate problems with the geometry). Statistics for the datasets are given in Table 1.

Table 1. Size, average, and standard deviation of the train, dev, and test sets.

Dataset	Size	Average (kcal/mol)	St. dev. (kcal/mol)
Train	138326	1.660	0.988
Dev	4970	1.674	0.926
Test	4974	1.646	0.868

Extension of prior approaches

The contribution of this project is twofold: to produce a dataset of molecular dimers, and to create a neural network capable of estimating BSSE. A direct comparison between gCP and a neural network trained on my database is deceptive because the neural network enjoys the advantage of a much larger and more diverse training set. Therefore, I reimplemented gCP in C and wrote an Adam optimization driver for gCP. The gCP correction takes substantially more time to run than a neural network, so the gCP model was trained for 3 epochs. After each epoch, the parameters were saved and the performance of those parameters was recorded. Parameter 2 was prevented from going below 0.001 due to numerical instability issues.

Table 2. Comparison of performance and parameters for gCP before and after parameter optimization.

	Dev MAE (kcal/mol)	Parameters
gCP from ref. 3	0.5613	0.129000 1.152600 1.154900 1.176300
gCP, epoch 1	0.2362	0.284139 0.001000 1.171643 1.064862
gCP, epoch 2	0.4231	0.040156 0.002068 1.395729 0.814701
gCP, epoch 3	0.4347	0.054609 0.001000 1.678147 0.753614

Optimizing the gCP parameters improves performance from about 40% mean relative error down to 16%. All three sets of optimized parameters outperform the default values, even though the three sets of optimized parameters vary wildly. This observation suggests that the parameter optimization is operating on a flat surface. The gCP correction contains two additional sets of parameters which were estimated empirically and held fixed during the optimization. Optimizing over these parameters as well may introduce some contours on the surface which allows for convergence to some minimum.

Code for this section can be found in the gCP folder.

Neural network architecture and discussion

Message-passing neural networks (MPNNs) are a subset of graph neural networks (Figure 1). Messages are composed of some attributes describing an edge and its two vertices. For this project, the messages are a tuple containing a distance between a pair of atoms, one from each molecule in the dimer, (edge) and a one-hot encoding of the atomic number for each of the two atoms (node). To limit the number of messages per dimer, all atom pairs further than 9 Angstroms apart were neglected.

The node block updates the vertex embeddings using the messages, and the global block embeds the messages. The embedded messages are now in a form which is independent of the size of the molecule and which can be used as input for later layers of a neural network. The design of MPNNs can be split into four parts: the content of the message (edge block), the vertex embeddings (node block), the process of embedding and pooling all the messages for each graph (global block), and the design of deeper layers after the global block.

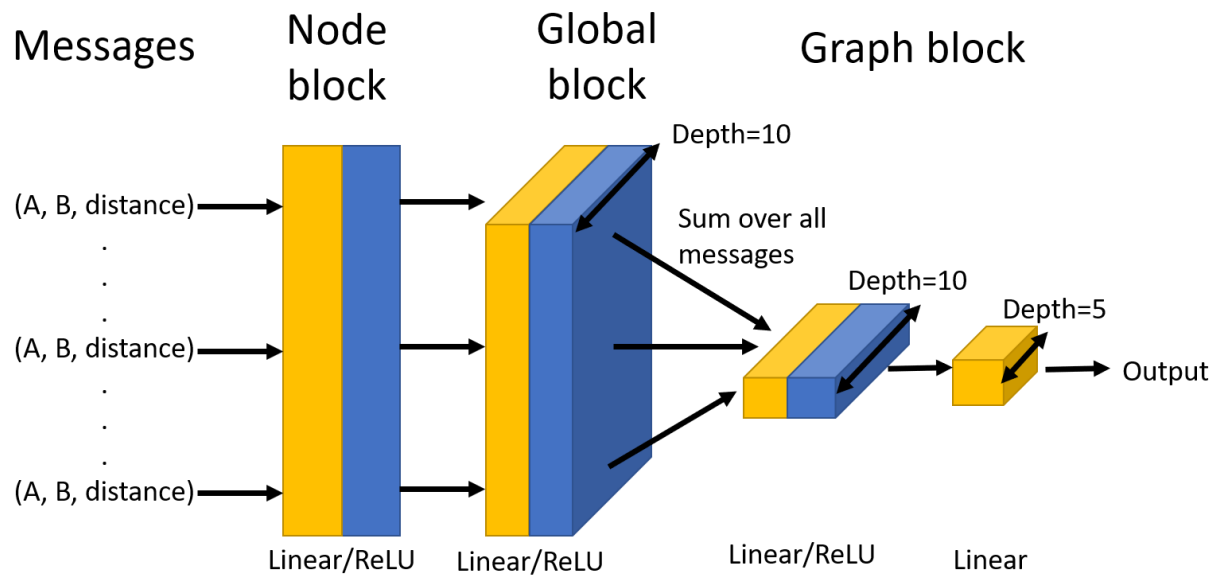

Figure 1. The base MPNN model. The messages are composed of the atomic numbers for the two atoms (A and B) and the distance between them. Depth indicates the length of the embedding vector.

The base model MPNN is shown in Figure 1. It is meant to be the simplest implementation of a MPNN. The node block is a single linear/ReLU layer, which feeds into the global block. The global block embeds each message into a 10-dimensional vector via a single ReLU layer. All the messages for each molecule are added up, producing one 10-dimensional vector per molecule. I denote all layers after this step to be the "graph block," since they operate on vector embeddings of the entire graph. In the base model, the graph block is one last ReLU layer, followed by a linear layer which reduces the vector to a scalar, the predicted energy.

It should not be surprising that variance for these models is low because the train, dev, and test sets are produced in the same way. While it is of interest to eventually test this model on molecules outside this test set, that transfer learning problem remains a future direction. This project focuses instead on minimizing bias through adjusting the network architecture. The base model was methodically augmented, and results are reported in Table 1. No model was optimized for more than 200 epochs, because the cost had converged to three decimal places by then and further optimization was not worth the time.

Table 3. Results and descriptions of the neural networks trained for this project. Mean absolute error (MAE) is given for the train set and for the dev set in kcal/mol and is taken from the epoch with the lowest dev MAE.

#	Description	Train MAE	Dev MAE	Epoch
1	Base model	0.2400	0.2098	149
2	Added additional Linear/ReLU layer in node block	0.2344	0.2049	115
3	Added additional Linear/ReLU layer in global block	0.2394	0.2073	178
4	Doubled depth of embedding vectors (10,10,5 to 20,20,10)	0.2261	0.1917	193
5	Added Linear/ReLU layer (depth 10) to start of graph block	0.2504	0.2211	166
6	Replaced ReLU activations with sigmoid	0.2433	0.2147	125
7	Added Linear/ReLU layer to node block and to global block	0.2342	0.2020	190
8	Took model 5, added Linear/ReLU layer to global block	0.2267	0.1925	197
9	Took model 4, added Linear/ReLU layer to global block	0.2234	0.1898	187
10	2 Linear/ReLU layers in node, global, and graph blocks	0.2268	0.1940	198
11	Added 2 additional Linear/ReLU layers to global block	0.2323	0.1976	187
12	Changed depth of embedding vectors to (30,20,10)	0.2146	0.1798	154
13	Took model 3, replaced ReLU activations with sigmoid	0.2449	0.2139	197
14	Took model 12, added Linear/ReLU layer to global block	0.2181	0.1823	176
15	Took model 3, replaced ReLU activations with tanh	0.2409	0.2101	166

For all models, the train MAE is higher than the dev MAE. This is because the train set has a higher standard deviation than the dev set. The increased spread of the data (and likely a few outliers) decreases performance on the train set relative to the dev set. In general, all models have a (Train MAE – Dev MAE) of about 0.03 kcal/mol, which indicates that variance is low. Model 1 is the base model shown in Figure 1, and models 2-6 alter one component of the design of model 1.

Models 7-15 combine the elements tested in models 2-6: increasing the depth of a particular block, increasing the size of the vector embedding, and changing the activation functions. Of these elements, adding a layer to the node block or global block and increasing the size of the embedding

vector were most effective (see models 2, 3, and 4), so those elements were incorporated into all the models in the next generation. In general, adding layers seems to have a marginal benefit (compare models 8 and 9 to models 5 and 4, respectively) and changing the activation function from a ReLU worsens performance (models 13 and 15).

The most important component of neural network design for this project is the depth of the embedding vectors. Embedding the messages into very long vectors produced the two most accurate models (12 and 14). Interestingly, adding additional layers onto models with large embedding vectors did little to improve performance (compare model 4 to model 9 and model 12 to model 14). The success of these deep embeddings may be explained by examining the functional form of gCP. The contribution of a single bond in gCP (analogous to a single message in the MPNN) is a function of the distance r, given as $\mathrm{f(r)} = e^{-\alpha r} \Big/ \sqrt{s_{ab} \cdot N_b}$, where α is a parameter, s_{ab} is an overlap integral of slater type orbitals centered on the atoms a and b, and N_b is a parameter based on atom b. Expressing a complicated function like this using a series of linear/ReLU layers on a vector would require either a very large number of layers or a very long vector, which explains why adding layers to the global block and increasing the size of the embedding vector seem to improve performance.

When training neural networks, one tends to reach a point where the model begins to overfit the training set and dev set performance declines. Many of the models here achieved their best performance on the dev set within the last 20 epochs of training, which indicates that there may be some small benefit from training those models further because they have not yet begun overfitting the dev set. Given that the epoch training costs were converged to about 0.001, it is unlikely that the MAE for any of those models would decline by more than 0.01 kcal/mol. Even with further training, these models would be unlikely to exceed the performance of models 12 and 14. Therefore, models 12 and 14 were selected to be used on the train set and compared against the performance of the current state-of-the-art (gCP).

Table 4. Test set performance in mean absolute error.

Model	Test MAE (kcal/mol)
gCP, from ref. 3	0.5853
gCP, epoch 1	0.2387
Model 12	0.1672
Model 14	0.1710

The optimized gCP parameters perform much better on the test set than those given in ref. 3, likely due to the advantage of training and testing on the same distribution. The comparison between the neural networks and the re-trained gCP model is more pertinent. Models 12 and 14, which have quite simple architectures and relatively few parameters, outperform the state-of-the-art by about 5% mean relative error, which corresponds to about a 33% reduction in error. In short, neural networks can significantly outperform the current state-of-the-art empirical corrections for estimating BSSE.

Code for this section can be found in the MPNN folder.

References:

1. S. F. Boys, F. Bernardi. *Mol. Phys.* **1970,** *19*, 553-566.
2. F. Jensen. *J. Chem. Theo. Comp.* **2010,** *6,* 100-106.
3. H. Kruse. S. Grimme, *J. Chem. Phys.* **2012,** *136,* 154101.
4. D. Weininger. *J. Chem. Inf. and Comp. Sci.* **1987,** *28,* 31-36.
5. T. Fink, J.-L. Reymond, *Angew. Chem. Int. Ed.* **2005,** *44*, 1504-1508.
6. L. Blum, J.-L. Reymond. *J. Am. Chem. Soc.* **2009,** *131*¸ 8732-8733.
7. L. Martinez, R. Andrade, E. G. Birgin, J. M. Martinez. *J. Comp. Chem.* **2009,** *30,* 2157-2164.
8. D.A. Case, I.Y. Ben-Shalom, S.R. Brozell, D.S. Cerutti, T.E. Cheatham, III, V.W.D. Cruzeiro, T.A. Darden, R. E. Duke, D. Ghoreishi, M. K. Gilson, H. Gohlke, A. W. Goetz, D. Greene, R. Harris, N. Homeyer, Y. Huang, S. Izadi, A. Kovalenko, T. Kurtzman, T.S. Lee, S. LeGrand, P. Li, C. Lin, J. Liu, T. Luchko, R. Luo, D.J. Mermelstein, K.M. Merz, Y. Miao, G. Monard, C. Nguyen, H. Nguyen, I. Omelyan, A. Onufriev, F. Pan, R. Qi, D.R. Roe, A. Roitberg, C. Sagui, S. Schott-Verdugo, J. Shen, C.L. Simmerling, J. Smith, R. Salomon, Ferrer, J. Swails, R.C. Walker, J. Wang, H. Wei, R.M. Wolf, X. Wu, L. Xiao, D.M. York, P.A. Kollman (2018), AMBER 2018, University of California, San Francisco.
9. H. Tatewaki, Y. Sakai, S. Huzinaga. *J. Comp. Chem.* **1981,** *2,* 278-286.
10. I. Ufimtsev, T. Martinez. *J. Chem. Theo. Comp.* **2009,** *5,* 2619.
11. A. Titov, I. Ufimtsev, N. Luehr, T. Martinez, *J. Chem. Theo. Comp.* **2013,** *9,* 213.