Scalar Coupling Prediction by LightGBM Powered Model

CHE696 FINAL PROJECT

Atsunori Kaneshige
College of Pharmacy
University of Michigan
Ann Arbor, MI
atsunori@umich.edu

April 20, 2019

PROJECT SUMMARY

In this project, I aimed to predict scalar coupling constant by using an idea presented in a Kaggle competition, *Prediction Molecular Properties* [1]. Because calculating scalar coupling constant by current quantum mechanics is expensive and time consuming, a faster and cheaper way to calculate the constant is quite useful in a variety of fields such as drug discovery research. Understanding of how to predict scalar coupling constant, feature engineering, and data preprocessing yield a dataset ready for training a model. I make a comparison of 12 machine learning algorithms to determine the algorithm of choice. The comparison indicates that Light Gradient Boosting Machine [2] has the best performance with regard to the prediction accuracy and calculation speed. The generated model can predict scalar coupling constant of any small molecule with high accuracy, and I believe that the predicted scaler coupling constants can be used as features of molecules in drug discovery activities, such as virtual high throughput screening.

Keywords LightGBM · Scalar Coupling Constant · J Coupling · Virtual Screening

1 Introduction

Compositions of atoms in molecules and their three dimensional structures can explain a great part of their chemical and physical properties. For example, a small molecule drug in many cases has a dominating structural form. The molecule structure and properly arranged atoms in the drug plays an important role in interacting its target protein. This protein also consists of variety of amino acids that have defined chemical structures.

These chemical structures can be represented by consisting atoms and their scaler coupling constants. In other words, a set of atoms and their scalar coupling constants, and their topology would back-produce a representing structure of a molecule. Current state-of-the-art quantum mechanics gives scalar coupling constants with an excellent accuracy and people can use these calculated constants to gain structural insights for their researches. However, the calculation is very expensive, taking days or weeks per molecule. The advent of machine learning potentially gives us a mean to alleviate this drawback while keeping the accuracy.

If we have access to accurate scalar coupling constants quicker and cheaper by a trained model through machine learning, we can obtain more structural insights quicker and the predicted coupling constants can be used as features to describe the structures of molecules.

2 Methods

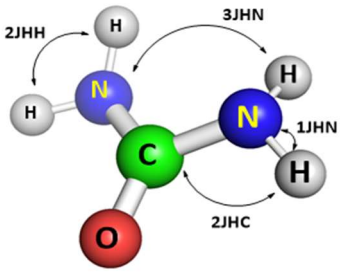
In this project, I used data that contains structure, bond type, and scalar coupling information of over 130,000 small molecules from a Kaggle competition, *Prediction Molecular Properties* [1]. The approach to predict scalar coupling constant is an idea of geometry and pattern matching among molecules, which I will explain in the following sections.

2.1 Data Source

A Kaggle competition, *Prediction Molecular Properties* [1], provides over 130,000 molecules' data of potential energy, dipole moment, magnetic shielding, mulliken charges, structure, scalar coupling constant, and bond type. Among these, the approach in this project only uses structure, bond type, and scalar coupling data. The train dataset contains 4,658,147 scalar coupling observations of 85,003 unique molecules, and the test dataset contains 2,505,542 scalar coupling observations of 45,772 unique molecules.

2.2 Scalar Coupling Constant and Coupling Types

Figure 1. Coupling Types



Scalar coupling constant known as J coupling is an indirect interaction between the nuclear spins of 2 atoms in a magnetic field**.** Dihedral angles between two interacting atoms, electron negativity of atoms, hybridization of atoms, and ring strain are main properties that affect J coupling. The molecules in the dataset contains only the atoms: carbon (C), hydrogen (H), nitrogen (N), fluorine (F), and oxygen (O). There are 8 different types of scalar coupling: 1JHC, 1JHN, 2JHH, 2JHC, 2JHN, 3JHH, 3JHC, 3JHN (Figure 1). Fluorine coupling was not represented in this dataset. The number that comes before the J in the J coupling

types (1J, 2J, 3J) denotes the number of bonds between the atoms that are coupling. So 1J, 2J, 3J coupling will have 1, 2, and 3 bonds between the atoms, respectively.

2.3 Approach

Over 2,800 teams from 84 countries precipitated in this Kaggle competition and submitted over 40,000 proposed solutions. From top ranked unique solutions, I got an inspiration to use geometry and pattern matching for the scalar coupling constant prediction [3]. The hypothesis behind the approach in this project is the following.

1. If we have two similar sets of atoms with the same distances between them and the same coupling types - the scalar coupling constant should be very close.

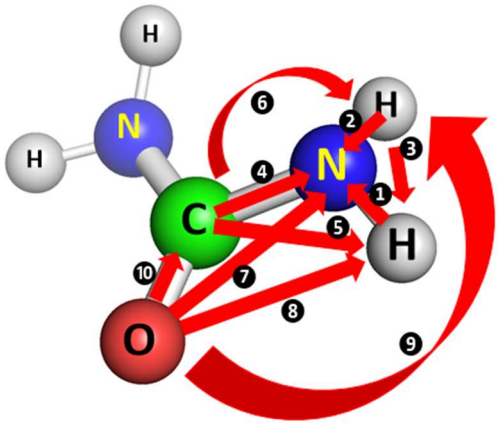
2. Closer atoms to the pair of atoms under prediction have higher influence on scalar coupling constant then those with further away.

Each atom is described with 3 cartesian coordinates. Here, a problem in this approach is that each coupling pair is located in a different point in space and two similar coupling sets would have very different X,Y,Z coordinates.

To alleviate this problem, I used the following 5 step system, instead of using given X, Y, Z coordinates as data features.

Step 1: Take each pair of atoms as two first core atoms. **Step 2:** Calculate the center between the pair. **Step 3:** Find all n-nearest atoms to the center (excluding the first two atoms). **Step 4:** Take closest atoms from step3 and sort the order atoms by distance. **Step 5:** Calculate the distances of all combination of atoms starting from core atoms (Figure 2).

Figure 2. Distances among atoms



By using this geometry/distance representation of atoms in a particular coupling in a molecule, each coupling pair can be described the distances among all atoms in a molecule. This representation is stable to rotation and translation. And it's suitable for pattern-matching. Now, we can take a sequence of surrounding atoms around core atoms, and describe the environment of the core atoms by distances and atom type such as hydrogen, carbon, and oxygen. We can check the same atom pattern and find similar distance pattern, and then guess a scalar coupling constant. In the case of 1JHN coupling in urea shown in figure 2, hydrogen and nitrogen that form the N-H bond are defined as core atoms (**Step 1**). From the center (**Step 2**) in the middle of N-H bond, the distances to nearest atoms are calculated (**Step 3**) and ordered the surrounding atoms from the closest (**Step 4**). Then, the algorithm calculates the distances of all atoms including core atoms, by following the indicated order, starting from core hydrogen and core nitrogen (**Step 5**). The obtained distances are used as features that belong to the numpy array of the 1JHN of interest.

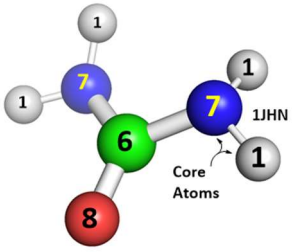
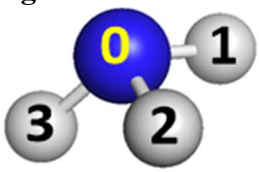


Figure 3. Surrounding atoms as features

In addition to the distances among atoms, all atoms are converted to atomic numbers as shown in figure 3, and the atomic numbers of atoms except core atoms are also used as features that belong to the same numpy array of 1JHN of interest. In the end, the 1JHN of interest has atomic numbers of surrounding atoms and distances among atoms as its feature set in the form of an numpy array.

2.4 Data Preprocessing

Figure 4. Ammonia



Here, I describe the data preprocessing by using a simple molecule, ammonia. Table 1 is a raw data and has molecule IDs, atoms, coupling types, and scalar coupling constants. The atoms of ammonia are indexed as indicated in figure 4. Ammonia has its ID and consists of 3 hydrogens and 1 nitrogen. There are total 6 pairs (1-0, 1-2, 1-3, 2-0, 2-3, 3-0), each of which has own coupling type and scalar coupling constant.

Table 1. Raw train data from Kaggle

id	molecule_index	atom_index_0	atom_index_1	type	scalar_coupling_constant
10	2	1	0	1JHN	32.688900
11	2	1	2	2JHH	-11.186600
12	2	1	3	2JHH	-11.175700
13	2	2	0	1JHN	32.689098
14	2	2	3	2JHH	-11.175800
15	2	3	0	1JHN	32.690498

Table 2. Structure data from Kaggle

	molecule_index	atom_index	atom	x	y	z
5	2	0	N	-0.040426	1.024108	0.062564
6	2	1	H	0.017257	0.012545	-0.027377
7	2	2	H	0.915789	1.358745	-0.028758
8	2	3	H	-0.520278	1.343532	-0.775543

Table 3. Preprocessed training data

```
X_train:
[[1.        1.        0.        0.        0.        1.01719   1.6185228
  1.0171872 1.6187098 1.0172079 1.6187056 0.        0.        0.
  0.        0.        0.        0.        0.        0.        0.
  0.        0.       ]
 [1.        1.        0.        0.        0.        1.0171872 1.6185228
  1.01719   1.6187056 1.0172079 1.6187098 0.        0.        0.
  0.        0.        0.        0.        0.        0.        0.
  0.        0.       ]
 [1.        1.        0.        0.        0.        1.0172079 1.6187056
  1.0171872 1.6187098 1.01719   1.6185228 0.        0.        0.
  0.        0.        0.        0.        0.        0.        0.
  0.        0.       ]]


y_train:
[32.6889 32.6891 32.6905]
```

Each atom has an atom type and its X, Y, Z coordinates (table 2). The data in these two tables are combined and processed based on the approach explained in the previous section, yielding the final data structure shown in table 3. Each array represents a coupling.

Table 3 shows only 1JHN couplings. Because ammonia has 3 1JHN couplings, there are 3 numpy array. The first 5 elements in an array are the information of surrounding atoms of the coupling of interest. Because a pair, N-H, has only 2 hydrogens as surrounding atoms, the first 2 elements are filled with 1 (H) and other 3 elements are filled with zero. From the 6th element in an array, values of distances are stored. Again, because ammonia has total 6

combinations, there are 6 distances in an array starting from 6th element. The same applies to all atom pairs of all other coupling types, namely 1JHN, 1JHC, 2JHH, 2JHN, 2JHC, 3JHH, 3JHN, and 3JHC, in other molecules.

3 Results

After data preprocessing, total 12 regression algorithms were trained with the processed data. Among 12 machine learning algorithms, Light Gradient Boosting Machine (LightGBM) yielded the best performance in terms of accuracy and computational speed for the scalar coupling constant prediction.

3.1 Mean Absolute Error Comparison with 12 algorithms

Table 4. Algorithm Comparison with 1JHN

Algorithm	Mean Absolute Error (log)
KMeans	3.852
DBSCAN	4.378
Lasso	1.338
Ridge	0.506
ElasticNet	1.898
MLPRegressor	-0.795
KerasRegressor	0.323
DecisionTreeRegressor	0.022
RandomForestRegressor	-0.761
GradientBoostingRegressor	-0.795
XGBRegressor	-0.969
LGBMRegressor	**-0.960**

In this project, I tried 12 machine learning algorithms to decide which algorithm to choose. Here, I only used 1JHN coupling type to test. Note that I used default settings for these algorithms and did not do careful tunings of hyperparameters. Table 4 shows type of algorithm and natural logarithm of mean absolute error, indicating that the smaller the value is, the more accurate the prediction is. First, I tried two of the most popular clustering (unsupervised) algorithms, KMeans and DBSCAN. As expected, both of them performed poorly in predicting 1JHN coupling, probably because of the characteristics of KMeans and DBSCAN. For example, KMeans needs the number of clusters hidden in the dataset (n_cluster = 2) and do not work well over clusters with different densities. The rest of algorithms are categorized in supervised machine learning algorithm. Next, I tried Logistic regression with regularization, namely Ridge (L2, alpha = 0.1), Lasso (L1, alpha = 0.1), and ElasticNet (L1 and L2, alpha = 0.1, l1_ratio = 0.5), yielding results with poor accuracy. DecisionTreeRegressor (max_depth = 9) gave a relatively good result without tuning much. Then I went to fancier algorithms. Multilayer Perceptron (MLP) and Keras. MLP is a class of feedforward artificial neural network. Keras is a polular library for deep learning and makes building deep learning a lot easier and quicker. Usually, the more complex an algorithm becomes, the more hyperparameters we have to tune. Finally I tested ensemble methods. RandomForest uses bagging strategy and showed a good prediction. In contrast, the bottom two algorithms use boosting strategy. Both eXtreme Gradient Boosting (XGB) [4] and Light Gradient Boosting Machine (LightGBM) [2] are built upon Gradient Boosting algorithm and the computational speed and efficiency have been significantly

improved. Among these ensemble methods, LightGBM algorithm yielded the best performance with regard to accuracy and computational speed.

3.2 LightGBM

Table 5. Hyperparameters

#Train_Set	#CV_split	#Feature
> 4,658,000	3	23

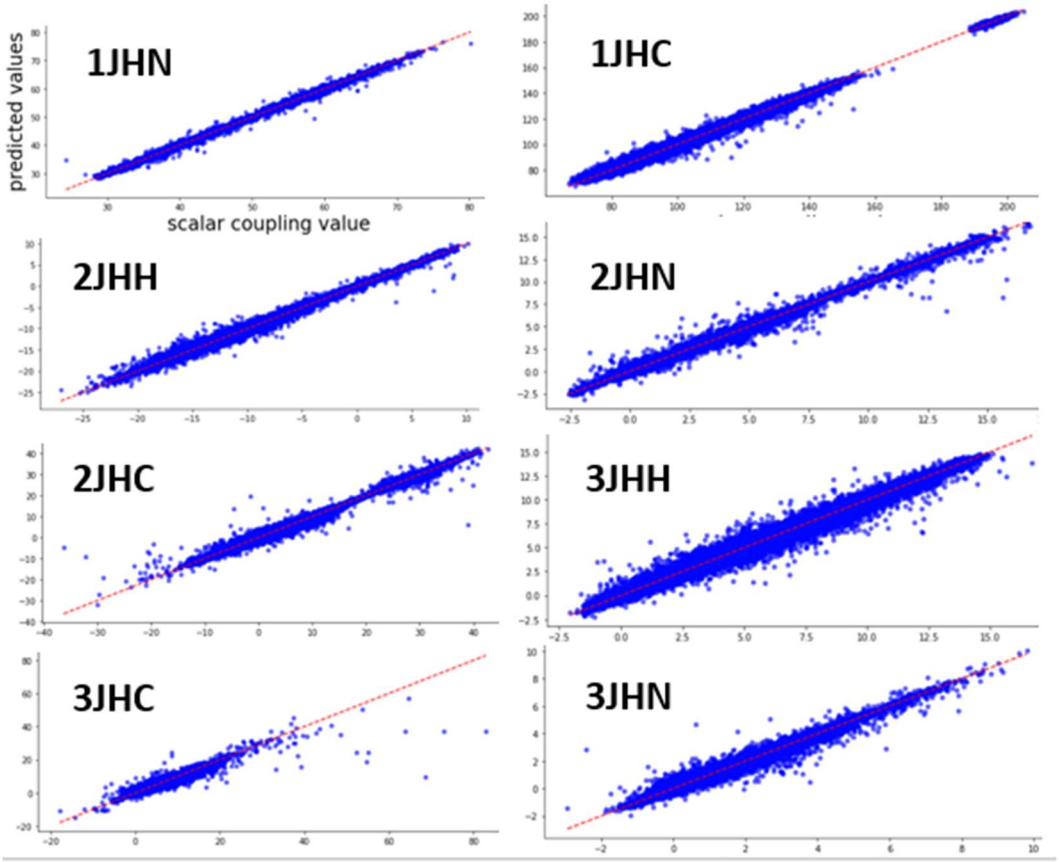
Parameters	settings
objective	'regression'
metric	'mae'
boosting_type	'gbdt'
learning_rate	**0.2**
num_leaves	128
max_depth	**9**
subsample_freq	1
subsample	0.9
reg_alpha	0.1
reg_lambda	0.3
colsample_bytree	1.0
n_estimaters	**1500**

LightGBM is a gradient boosting framework [4] that uses tree based algorithms. LightGBM is designed to be distributed and efficient with its advantages. For example, faster training speed and higher efficiency. Lower memory usage. Better accuracy. Support of parallel and GPU leaning. Also, capability of handling large-scale data. Gradient boosting algorithms like LightGBM and XGBoost have many parameters and just a part of them are shown in table 5. But, there are rules of thumb for configuring them [5,6,7]. I performed k-fold cross-validation (k = 3) with each coupling type, starting from 1JHN and ending with 3JHN. The true scalar coupling values and the predicted values are plotted as shown in Figure 5. Also, obtained mean absolute errors of training dataset and validation dataset are shown in Table 6 . Over all, the model predicting the 3JHN coupling constants performed the best with a validation score of -2.18, and the model predicting the 1JHC coupling constants performed the worst with a validation score of -0.24. Considering the fact that I did not tune hyperparameters intensively and that the best publicly available score on Kaggle is -3.24 [1], the obtained result is satisfying.

Table 6. Mean Absolute Errors of All Coupling Types

Coupling_Type	CV_Train_MAE	CV_Validation_MAE (log)	Target_value_Mean	Target_value_STD
1JHN	0.119	0.385 (-0.96)	47.528	10.925
1JHC	0.514	0.778 (-0.24)	94.962	18.287
2JHH	0.092	0.180 (-1.71)	-10.271	3.993
2JHN	0.056	0.147 (-1.92)	3.109	3.667
2JHC	0.233	0.308 (-1.18)	-0.276	10.550
3JHH	0.113	0.176 (-1.73)	4.772	3.712
3JHC	0.275	0.340 (-1.08)	3.687	3.076
3JHN	0.042	0.113 (-2.18)	0.994	1.319

Figure 5. Mean Absolute Errors of Cross-Validation Training vs True Values



4 Conclusion

In this project, I aimed to predict scalar coupling constant by using an idea presented in a Kaggle competition, *Prediction Molecular Properties* [1]. Because calculating scalar coupling constant by current quantum mechanics is expensive and time consuming, a faster and cheaper access to it is quite useful in a variety of fields such as drug discovery research. The model with Light Gradient Boosting Machine performed the best with regard to the prediction accuracy and computation speed among 12 algorithms tested, yielding very competitive scores. The generated model can predict scalar coupling constant of any small molecule with high accuracy, and the predicted scaler coupling constant can be used as new features that describe molecules in drug discovery activities, such as virtual high throughput screening.

Potential avenues to improve the accuracy would be 1) to tune hyperparameters further, 2) to include more surrounding atoms, 3) to add other features such as dipolar moments, mulliken carges, or potential energy, 4) to use categorical features for atom types (i.e., one hot encoding), and 5) to use other tree libraries such as CatBoost [8].

References

[1] Predicting Molecular Properties (https://www.kaggle.com/c/champs-scalar-coupling)

[2] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. and Liu, T.Y., 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems* (pp. 3146-3154).

[3] Distance – is all you need (https://www.kaggle.com/criskiev/distance-is-all-you-need-lb-1-481)

[4] Chen, T., He, T., Benesty, M., Khotilovich, V. and Tang, Y., 2015. Xgboost: extreme gradient boosting. *R package version 0.4-2*, pp.1-4.

[5] Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp.1189-1232.

[6] Friedman, J.H., 2002. Stochastic gradient boosting. *Computational statistics & data analysis*, *38*(4), pp.367-378.

[7] Hastie, T., Tibshirani, R. and Friedman, J., 2009. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

[8] CatBoost (https://catboost.ai/)