Capstone Report: Molecule Coupling Constant Prediction with Deep Learning

Yan Gao[1, *]

[1]*Daniel Felix Ritchie School of Engineering & Computer Science, University of Denver, Denver 80210, America*

(Dated: August 14, 2019)

An process of how to solve the small group problem is given. The method is implemented on a molecule dataset, which tries to use the properties of atoms in molecules to predict the scaler coupling constant in each pair of atoms. Both Spark and keras are implemented as I focus my attention on feature engineering and deep learning part. Main problem here is to use Spark to build features in graph data and convert the data to table data so that I can use it to train the model.

PACS numbers:

I. INTRODUCTION

The small graphs problem is really a interesting topic to analyze. It's not like a large graph with billions of nodes and incredible number of relationships. It has its own unique properties. The large dataset could be divided into considerable number of groups by finite number of relationships. So for each individual group, the size is much smaller and easy to evaluate how the nodes can talk with each other.

There are several parts in small group people interesting in, like how to define the relationship between different nodes, how to evaluate the strength of the relationship between two random nodes in a graph and how to quantify the influence of a graph to any nodes in it. Take the Facebook relation for example. This could be explained as exploring the true relation of two user in same group in Facebook, how close they are based on their relationship and how group members in a group will influence on others according to their relationships.

The small graph problem also has a particular field that need to take into consideration. That's how should we convert the graph database [1] into a rational one [2] that could be feed into the model. This is because if we want to quantify the influence of graph structure, we need to let nodes in graph talk with each other through their relationships. After we finished the evaluation part, we need to export the data to a 2 dimensional table so we could feed it to our model.

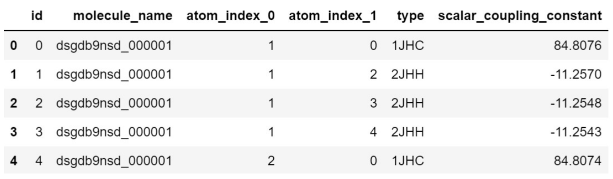
One other problem in all feature engineering processes is to add hidden effect. There are some nodes in the graph has unpublished information or the relationship information is unknown. This hidden effect [3] is quite vague but could be really import for decrease variance and making the probability distribution of our goal be precise.

I choose a molecule dataset which has the information of atom and some major coupling type between atoms. This is typically a small group problem as each molecule could be treated as a small graph with its atoms as nodes. I want to address the importance of those properties I mentioned above and try to use deep learning model to solve it.

*Electronic address: Yan.Gao@du.edu

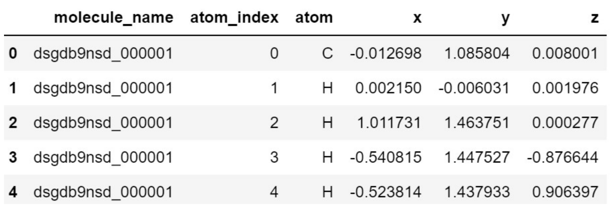
II. DATASET DESCRIPTION

This is a Kaggle competition dataset [4]. This is a typical relational dataset which puts all the nodes information in one CSV file and all relationships in the other. The train and test file contains the pairs relation of atoms in different molecules. The major information has:



	id	molecule_name	atom_index_0	atom_index_1	type	scalar_coupling_constant
0	0	dsgdb9nsd_000001	1	0	1JHC	84.8076
1	1	dsgdb9nsd_000001	1	2	2JHH	-11.2570
2	2	dsgdb9nsd_000001	1	3	2JHH	-11.2548
3	3	dsgdb9nsd_000001	1	4	2JHH	-11.2543
4	4	dsgdb9nsd_000001	2	0	1JHC	84.8074

FIG. 1: The fields in train data.



	molecule_name	atom_index	atom	x	y	z
0	dsgdb9nsd_000001	0	C	-0.012698	1.085804	0.008001
1	dsgdb9nsd_000001	1	H	0.002150	-0.006031	0.001976
2	dsgdb9nsd_000001	2	H	1.011731	1.463751	0.000277
3	dsgdb9nsd_000001	3	H	-0.540815	1.447527	-0.876644
4	dsgdb9nsd_000001	4	H	-0.523814	1.437933	0.906397

FIG. 2: The fields in structure data.

- molecule_name: the id of each molecule. Several atom may belongs to same molecule and has same molecule_name.
- atom_index: the index of atom which is unique in each molecule. It will occur repeatedly in different molecules. By using this we could look up the information of atom in the nodes file but the index itself dose not indicate any physical property.
- type: the coupling type between two atoms pair. Each atom pair should have a coupling type.
- x, y, z: the spatial position of each atom.
- atom: the element of the atom.

These are basically all the predictors I have. Normally this is the most common information we have for measure the properties of the atom. The response variable is the scaler coupling constant which only given in the train data. So this is a continuous response. Most of predictors are categorical. This means that they could be easy to evaluate.

There are methods like EDA [5] being use to predict such problems. I use feature engineering skill with my physics domain knowledge to add more predictors to the data. Otherwise the 5 predictors should not be enough for billions of data and will cause a high variance.

III. MAIN METHOD

Firstly I need to find the distribution of the scaler coupling constant in the train data. It should tell how imbalance the response variable is. Then I should compare the distribution of the response with different categorical predictors, to check if there are any predictors that contribute more to the response. Also if there are any pattern that does not match any predictors, it indicates that there should be some hidden effect that influence the response. These unknown information should be denote somehow in the feature engineering part.

The reason we should add unknown effects is that it could be a influential point in the graph while it could be a shift in the table data. It's not like the constant intercept we put into the model which denotes all the general influence caused by environment. These unknown effects could influence part of data and modify the probability distribution a bit, causing high variance and even outliers.

Then it comes to feature engineering. More features are added to the data. With physics knowledge, I add the distance between each atom pair, the radius of each atom from the molecule mass point and the potential energy between each atom pair. This are obvious features. More feature are not so easy to find like the influence of each coupling type or element to just one atom or one coupling in the molecule. This is the way I intend to use to quantify the structure influence of molecule to the individual atom. Apache Spark could work as I assume.

Spark has a build in graphframe library [6] which could input nodes and relationships data to build a graph. Then I can use Pregel library [7] to send message between nodes according to different relationships. Once the effects is quantified, I can send them between nodes and recalculate them for overall influence. Spark module is built for parallel computing, so it should run fast as we could map the calculation part to several worker nodes by hashing the molecule_name(identical number for each molecule).

After all the features being extracted from the Spark and inserted into our train table data, I can use Python keras library to build a deep learning model and use data to train it. Mainly I need to find how many layers and the nodes in layers I should have, how should I choose my optimizers and estimators, and select the best number of epochs.

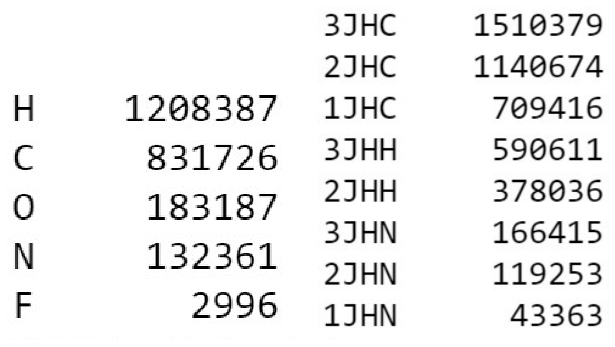
These are my main method to solve the small group problem. Now let's we look at how the application works.

IV. APPLICATION DETAIL

A. Exploratory Data Analysis

1. The graph statistics

The train dataset has 85003 molecules and each of them has atoms ranging from 5 to 30. So we need to train 85003 small graphs.



```
H    1208387
C     831726
O     183187
N     132361
F       2996
```

(a)

```
3JHC    1510379
2JHC    1140674
1JHC     709416
3JHH     590611
2JHH     378036
3JHN     166415
2JHN     119253
1JHN      43363
```

(b)

FIG. 3: (a) Number count of element in all atoms. (b) Number count of coupling types in the train data.

The total dataset contains 5 different atom's elements and their number are quite unbalanced. Another problem is that the coupling type given only include the coupling between hydrogen, carbon and nitrogen. All coupling to oxygen and fluorine are unknown. But it's obvious that even if I don't know it, it should have influence on other atoms in same molecule.

2. The response distribution

I am so convinced that the coupling type should be strongly related to the scaler coupling constant in the molecule. So a plot to show the distribution of the response and the average response according to coupling type should give us some inspire. From the FIG 5, when the response is low, the coupling type really matches. But with the increase of the response, there seems to be a shift between the patterns and the averaged values. So this shows that there should be some hidden effect in the molecule structure, which only exists in some of molecules and will influence the response dramatically. This effect, as I imagine, is the atom of oxygen or fluorine element and their corresponding coupling type with other atoms.

3. Graph structure

B. Model parameter setting and tuning

For the model feeding part, I choose to use keras deep learning library.

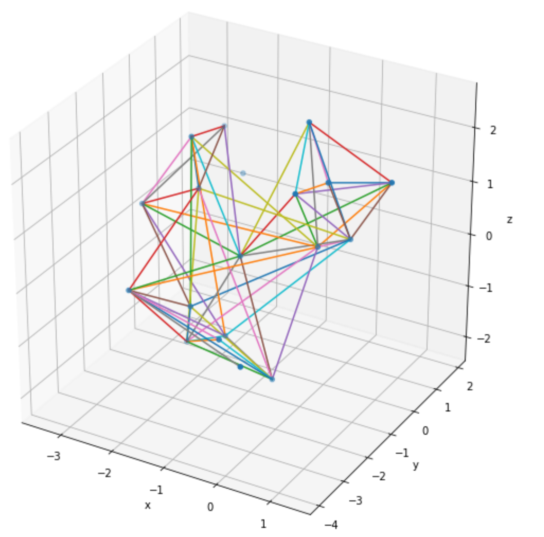


FIG. 4: This is a molecule drawn randomly from the train data. Each vertex means a atom. The line between different atoms mean the given scaler coupling type. From the figure, we can see that there are atoms that do not have any connections with others. This means the effect of the atom is unknown, which is part of the hidden effect.

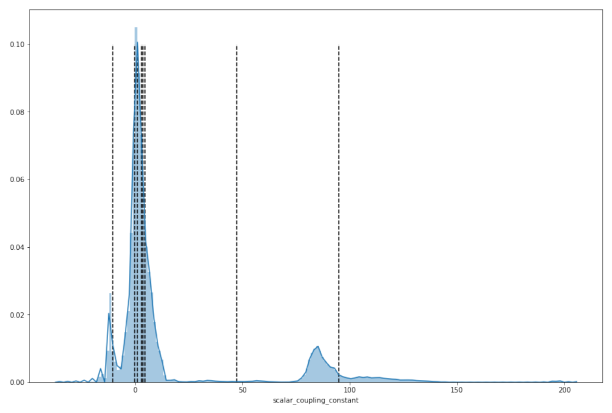


FIG. 5: This shows the distribution of the scaler coupling constant(response variable). The dashed lines denote the averaged coupling constants of different coupling type.

I set 5 layers of the neuron network. The first four layers are based on RELU activation with increasing number of neurons. And the last layer just has one output to give me a continuous response. There are many choices for the activation function. The RELU function only keep the positive values. Other choices like ELU, PRELU or CRELU may also work.

The mean squared error is set to be the loss optimizer. Also in this case, the ϵ-loss also make sense with classification. It's more wildly used in the SVM model.

I use k-fold validation to run the model several time and get the average of the result to find the best number of epochs to set for the final result. After the general search for good performance, I restrict the limit to the certain range and make a comparation of how good the performance for different number of epochs, shown in FIG 6.

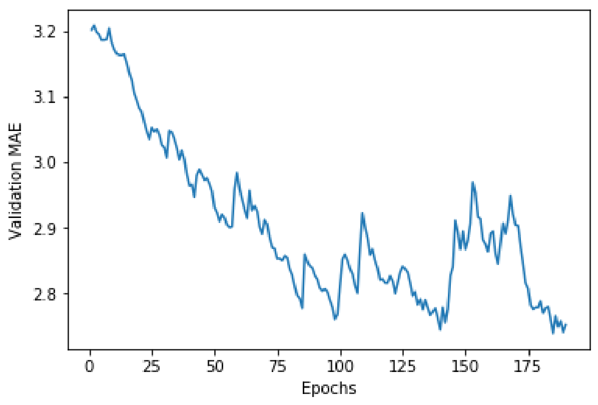


FIG. 6: This is the deep learning running result of 200 epochs with 4 folds. The one with lowest mae is what we expected.

C. Result score

Derived from the plot above, I finally choose the number of epoches to be 185. Then we can see the best performance of our model under this parameters:

log_mse_by_type	0.744
rank	1984

D. Result explanation

The evaluation function given by the competition is given as:

$$score = \frac{1}{T}\sum_{t=1}^{T}\log\left(\frac{1}{n_t}\sum_{i=1}^{n_t}|y_i - \hat{y}_i|\right) \quad (1)$$

where T is the number of scaler coupling types, n_t is the number of observations of type t, y_i is the actual scaler coupling constant for the observation and $\hat{y}_i$ is the predicted scalar coupling constant for the observation. So this score is basically the logged mean squared error averaged by the scaler coupling constant. The ideal score for this competition should be -20.7232. The negative score is caused by the log function. So only reaching a pretty high precision can I get a negative score. From the result, my result score is 185, which is still need some improvement.

V. ERROR ANALYSIS

I can't use Spark to build such a large graph above 3.5 GB. So some most important structure parameters are not actually implemented. Spark is built to be parallel, which should works well. But I only have one machine and can't make the best performance of it. To calculated with spark, I need to take my data apart to several smaller files so that I can run a smaller enough chunk at once. Separating data takes lots of time and even if I separated it successfully, running these small chunks also takes many times. So the best solution is to get a cluster of machines and run this in parallel system. Spark will automatically slices data into small fragments and send them to the work nodes. This saves lots of time. Otherwise I should find a better graph builder rather than Spark.

Without build graph and run it successfully with Spark, I can't extract many features as I should do. So only for those feature I build, there still be a lack of information for achieve a high accuracy model.

Another problem Spark has is that it can only create new parameters on nodes but can't change relationships and their already created fields. This is quite awkward as I need to e-valuate the relationship data(coupling between atoms). And if the already created fields can't be change, it could be inconvenient to update values in the corresponding field and make many information converted though the graph at the same time.

On the other hand, train the deep learning model with such a large dataset could be really slow as I need to find the best number of epochs. So I intended to train it on cloud at first. But there is no build in image of keras in the AWS. So although I can install and run keras in the AWS, I can't save the trained model. This means I still need to re-train my model each time I re-open my notebook instance on the cloud.

One more trouble is that I can use the GPU machine on AWS which is limited on my account. So only with a machine with powerful CPU still can't speed up the model training process.

VI. CONCLUSION

The small graph problem is not so easy to achieve. Compared with large connected graph of same size, the small group graph set actually requires less computing power. But in this condition, it's still quite overwhelmed my laptop. The deep learning model requires really flexible tuning ability, which need some insight of how to start and set the parameter-s. By tuning the parameters, the running speed would increase or decrease somehow. So in limited time, it really requires me to think how to set parameter properly and understand how it costs.

The result is not satisfying. So next step, I will try to use Neo4j dataset [8] to build the graph and query its properties. It's has more functionality to contain more nodes and relations at same time. And it can achieve fast querying by creating corresponding index. This can help me build more features to feed into the model.

Then I will try to implement this on Google Cloud to check if there is any better results.

[1] https://en.wikipedia.org/wiki/Graph_database
[2] https://en.wikipedia.org/wiki/Relational_database
[3] https://en.wikipedia.org/wiki/Hidden-variable_theory
[4] https://www.kaggle.com/c/champs-scalar-coupling
[5] https://www.kaggle.com/artgor/molecular-properties-eda-and-models
[6] https://databricks.com/blog/2016/03/03/introducing-graphframes.html
[7] https://www.qubole.com/blog/processing-hierarchical-data-using-spark-graphx-pregel-api/
[8] https://neo4j.com/