DFT with machine learning methods*

Yu-Chieh Chi†
Department of Scientific Computing, Florida State University, Tallahassee, Florida 32306, USA
(Dated: December 12, 2018)

In this project, I tried to implement some results from Faber's paper [1]. It use Machine learning to approximate some parameters from density functionals theory (DFT). Base on the Molecular structures and properties at the hybrid DFT level of theory come from the QM9 database [2] and include enthalpies and free energies of atomization, HOMO/LUMO energies and gap, dipole moment, polarizability, zero point vibrational energy, heat capacity, and the highest fundamental vibrational frequency. All molecular representations have been based on Coulomb matrix (CM). Regressors include linear models (Bayesian ridge regression (BR) and linear regression with elastic net regularization (EN)), random forest (RF). The results from BR method is very close to author's result. Because of the computation time is expensive, I did not calculate all results.

I. CODE IMPLEMENTATION

Please follow these steps to implement and eximize my final project code.

1. Install free software machine learning scikit-learn for the Python programming Langurage. Scikit-learn requires:
   - i Pthon ($\geq$ 2.7 or $\geq$ 3.4)
   - ii Numpy ($\geq$ 1.8.2)
   - iii Scipy ($\geq$ 0.13.3)
2. Download the Supplementary Materials at `https://drive.google.com/uc?id=0Bzn36Iqm8hZscHFJcVh5aC1mZFU&export=download`
   - (i) Unzip "Supplementary Materials.zip".
   - (ii) Go to the Supplementary Material folder, and decompress "(10, 30 50 70, 90)percentTest-10fold.gz", "(10, 30, 50, 70, 90)percentTest-nontest.gz", and "(10, 30, 50, 70, 90)percentTest-test.gz"
   - (iii) decompress "CM.gz"
3. Change the PATH.
   - (i) Go to "fold.py", and change the PATH on the line 6.
   - (ii) Go to "testBR.py", "testEN.py", and "testRF.py", and change the PATH on the line 13, 21, and 46.
4. Run "testBR.py" for Bayesian ridge (BR), and the "CM-BR.txt" file will be created. The final result will be inside the "CM-BR.txt".
5. Run "testEN.py" for Elastic Net regularization (EN), and the "CM-EN.txt" file will be created. The final result will be inside the "CM-EN.txt".
6. Run "testRF.py" for Random Forest (RF), and the "CM-RF.txt" file will be created. The final result will be inside the "CM-RF.txt".

Please let me know if above steps works ot not. I suggest you run Bayesian regression "testBR.py" first, because it takes shorts time than others.

II. MY PURPOSE

I saw Snyder's paper[3] and found that the content made me feel very kind. The work of this article is very clear from the goal, that is, the molecular information is input into ML (Machine Learning), and the various properties of the molecule are obtained after supervised learning. But the work of this article did not provide data set, which is technically difficult for me to implement. When I saw Faber's paper, I intend to start with this relatively simple article. Learn how machine learning is involved in microscopic theoretical chemistry research. Most of the methods in this article use a more traditional machine learning approach, namely ML that excludes deep learning.

In this project, I will briefly describe the work of Faber et al.[1] that I understand, give the implementation code for some of the results, and make a very simple conclusion.

Machine learning methods used by Faber et al. are four types: Bayesian Ridge Regression, Elastic Net, Random Forest, and Kernel Ridge Regression. Among them, the first two I understand as a linear fitting model with different types of penalty functions. The fourth I understand that the penalty term is introduced in the error function, so the parameterless fitting of the penalty parameter is also introduced in the kernel. It is much like a weighted average, except that the weight is determined by the kernel and the input vector being tested. Random Forest as a regression method I still don't understand now.

---

* A footnote to the article title

† yc16b@my.fsu.edu

All the work that is repeated is just the three lines in Table 3 from Faber et al[1]. But I think the repetition of these two lines gives me confidence that I believe in a thorough understanding of their work in the future.

III. PAPERS REVIEW

Most DFT studies solve kinetic energy from the Kohn-Sham equation. However, in principle, the theorem of DFT is not needed. If the kinetic energy of such electrons is known to be a function of density, a single equation can be solved to obtain accurate self-consistent density and extracted energy. There is no need to solve the KS equation, and the calculation bottleneck may solve the Poisson equation. Therefore, the DFT calculation will become much faster than it is now.

In Snyder's paper[3], they use ML to approximate the kinetic energy of a particle that precisely limits the particle in a one-dimensional box. Applying this method to real complex systems will enable highly accurate non-orbital density functional calculations and ab initio molecular dynamics simulations.

Behler [4] and Rupp [5] have recently adopted two methods. Try to approach the quantum mechanics solution directly without attracting DFT. In the first case, a single hidden layer neural network is used to approximate the energy and force of the helium melt configuration in order to accelerate molecular dynamics simulation. The second paper uses core regression (KRR) to infer the atomization energy of various molecules. In both cases, the physical symmetry is built into the input representation using hand-designed features (symmetric functions and Coulomb matrices, respectively). Subsequent papers have replaced KRR with neural networks.

Both studies used hand-design features with inherent limitations. Behler's work uses a representation that is clearly invariant to the isomorphism of the graph, but when applied to systems with more than three atoms, it is difficult to generalize to new components. The representation used in Rupp is not constant for graph isomorphism. Instead, this invariance must be learned by the downstream model through data set expansion.

IV. METHOD

A. Coulomb Matrix

The Coulomb matrix (CM) is a two-dimensional matrix. Each row or column of it represents an atom. The composition of the matrix is simple, that is, the Coulomb classic mutual exclusion between every two atoms. However, the atomic auto-rejection that does not actually exist on the diagonal of the matrix is taken care of here, so the following matrix is given:

$$M_{I,J} = \begin{cases} 0.5Z_I^{2.4} & \text{for} I = J \\ \frac{Z_I Z_J}{|\mathbf{R}_I - \mathbf{M}_J|} & \text{for} I \neq J \end{cases} \quad (1)$$

Here, off-diagonal elements correspond to the Coulomb repulsion between atoms I and J, while diagonal elements encode a polynomial fit of atomic energies to nuclear charge [5].

B. Data Set

Data set relevant to this project can be downloaded at `https://drive.google.com/uc?id=0Bzn36Iqm8hZscHFJcVh5aC1mZFU&export=download`. The link provides access to the following files: (i) Areadme file, (ii) names of the molecules that failed conversion from coordinates to rational SMILES strings for the MG representation, (iii) all representations used in this work, (iv) all splits used for training, validation and test set, and (v) properties of all molecules in the QM9 data set.

C. Bayesian Ridge Regresion

First of all, I use Bayesian Ridge Regresion (BR) as is implemented in scikit-learn. BR is a linear model with a L_2 penalty on the coefficients. Unlike ridge regression where the strength of that penalty is a regularization hyperparameter which must be set, in Bayesian ridge regression the optimal regularizer is estimated from the data[1].

D. Elastic Net Regularization

The generality of the elastic net regression is still preferable to either L_1 or L_2 regularization on its own. I just follow the regularization hyperparameter from the Faber's paper[1], for the weight penalty, elastic net has an additional hyperparameter l1_ratio to control the relative strength of the L1 and L2 weight penalties. I used the scikit-learn implementation and set l1_ratio = 0.8. I then did a hyperparameter search on regularizing the parameter in a base 10 logarithmic grid from 1e6 to 1.0.

E. Random Forest

I use RF as implemented in scikit-learn. RF regressors produce a value by averaging many individual decision trees fitted on randomly resampled sets of the training data. Each node in each decision tree is a threshold of one input feature. Early experiments did not reveal strong differences in performance based on the number of trees used (see Supporting Information for details), once a minimal number was reached.

Table 3. MAE on out-of-Sample Data of All Representations for All Regressors and Properties at ~118k (90%) Training Set Size[a]

		U_0, eV	ε_{HOMO}, eV	ε_{LUMO}, eV	$\Delta\varepsilon$, eV	μ, Debye	α, Bohr^3	ZPVE, eV	C_v, cal/molK	ω_1, cm^{-1}	NMMAE, arb. u.
EN	CM	0.9110	0.3380	0.6310	0.7220	0.844	1.330	0.02650	0.9060	131.00	0.4230
	BOB	0.6020	0.2830	0.5210	0.6140	0.763	1.200	0.02320	0.7000	81.40	0.3500
	BAML	0.2120	0.1860	0.2750	0.3390	0.686	0.793	0.01290	0.4390	60.40	0.2310
	ECFP4	3.680	0.2240	0.3440	0.3830	0.737	3.450	0.27000	1.5100	86.60	0.4620
	HDAD	0.0983	0.1390	0.2380	0.2780	0.563	0.437	0.00647	0.0876	94.20	0.1830
	HD	0.1920	0.2030	0.2990	0.3600	0.705	0.638	0.00949	0.1950	104.00	0.2360
	MARAD	0.1830	0.2220	0.3050	0.3910	0.707	0.698	0.00808	0.2060	108.00	0.2560
	mean	0.8400	0.2280	0.3730	0.4410	0.715	1.220	0.05090	0.5780	95.10	
BR	CM	0.9110	0.3380	0.6320	0.7230	0.844	1.330	0.02650	0.9070	131.00	0.4240
	BOB	0.5860	0.2790	0.5210	0.6140	0.761	1.140	0.02220	0.6840	80.90	0.3430
	BAML	0.2020	0.1830	0.2750	0.3390	0.685	0.785	0.01290	0.4440	60.40	0.2290
	ECFP4	3.6900	0.2240	0.3440	0.3830	0.737	3.450	0.27000	1.5100	86.70	0.4620
	HDAD	0.0614	0.1400	0.2380	0.2780	0.565	0.430	0.00318	0.0787	94.80	0.1820
	HD	0.1710	0.2030	0.2980	0.3590	0.705	0.633	0.00693	0.1900	104.00	0.2350
	MARAD	0.1710	0.1840	0.2570	0.3150	0.647	0.533	0.00854	0.2010	103.00	0.2260
	mean	0.8280	0.2210	0.3670	0.4300	0.706	1.190	0.05000	0.5740	94.50	
RF	CM	0.4310	0.2080	0.3020	0.3730	0.608	1.040	0.01990	0.7770	13.20	0.2390
	BOB	0.2020	0.1200	0.1370	0.1640	0.450	0.623	0.01110	0.4430	3.55	0.1420
	BAML	0.2000	0.1070	0.1180	0.1410	0.434	0.638	0.01320	0.4510	**2.71**	0.1410
	ECFP4	3.6600	0.1430	0.1450	0.1660	0.483	3.700	0.24200	1.5700	14.70	0.3490
	HDAD	1.4400	0.1160	0.1360	0.1560	0.454	1.710	0.05250	0.8950	3.45	0.1980
	HD	1.3900	0.1260	0.1390	0.1500	0.457	1.660	0.04970	0.8790	4.18	0.1970
	MARAD	0.2100	0.1780	0.2430	0.3110	0.607	0.676	0.01020	0.3110	19.40	0.1990
	mean	1.0800	0.1420	0.1740	0.2090	0.499	1.430	0.05690	0.7610	8.74	
KRR	CM	0.1280	0.1330	0.1830	0.2290	0.449	0.433	0.00480	0.1180	33.50	0.1360
	BOB	0.0667	0.0948	0.1220	0.1480	0.423	0.298	0.00364	0.0917	13.20	0.0981
	BAML	0.0519	0.0946	0.1210	0.1520	0.460	0.301	0.00331	0.0820	19.90	0.1050
	ECFP4	4.2500	0.1240	0.1330	0.1740	0.490	4.170	0.24800	1.8400	26.70	0.3830
	HDAD	**0.0251**	0.0662	0.0842	0.1070	0.334	0.175	**0.00191**	**0.0441**	23.10	0.0768
	HD	0.0644	0.0874	0.1130	0.1430	0.364	0.299	0.00316	0.0844	21.30	0.0935
	MARAD	0.0529	0.1030	0.1240	0.1630	0.468	0.343	0.00301	0.0758	21.30	0.1120
	mean	0.6620	0.1010	0.1260	0.1590	0.427	0.859	0.03830	0.3330	22.70	
GG	MG	0.0421	0.0567	0.0628	0.0877	0.247	**0.161**	0.00431	0.0837	6.22	0.0602
GC	MG	0.1500	**0.0549**	**0.0620**	**0.0869**	**0.101**	0.232	0.00966	0.0970	4.76	**0.0494**

[a]Regressors include linear regression with elastic net regularization (EN), Bayesian ridge regression (BR), random forest (RF), kernel ridge regression (KRR), and molecular graphs based neural networks (GG/GC). The best combination for each property is highlighted in bold. Additionally, the table contains mean MAE of representations for each property and regressor, normalized by MAD (see Table 4) and mean MAE (NMMAE) over all properties for each regressor/representation combination.

FIG. 1. Table 3 from Faber's paper[1]. I only repeat EN-CM, BR-CM, and RF-CM results.

V. IMPLEMENTATION RESULTS

Here, the BR data of all 13 properties of the CM feature vector are repeated. I only calculated the μ and did not calculate all properties for EN, and RF regression because of the time consume. The training process uses the Python package sklearn. The operation system is the latest version of macOS Majave 10.14.1, Python 2.7, Spyder 3.3.2. Hardware is 2.9 GHz Intel Core i7 CUP processor and with 16GB Memory.

Tables I lists the values from the Faber's paper, and the values I calculated. It can be seen that the error of most data does not exceed 3%. This should indicate that the program I am performing is correct. Most values of EN and RF are empty because the computational time is expensive. I need around 1.5 hours to get one value by using EN regression, and around 4.5 hours to get one value by using RF regression. Because the computational time is very expensive, I did not get all results from EN and RF regression.

VI. CONCLUSION

With ML on the DFT, I understand that there are at least two approaches now. One is to use ML directly as a functional [6], and the other approach is the practice of this article, which is to fit the DFT results with ML. The second approach is the one here.

If I choose the second path mentioned in the previous

TABLE I. Compare with Faber's[1] and my Result.

Regression		BR			EN			RF		
Parameters	Unit	Faber	My Result	Training Time	Faber	My Result	Training Time	Faber	My Result	Training Time
μ	Debye	0.844	0.84413750	178.04	0.844	0.84425038	6961.47	0.608	0.60061758	16636.60
α	Bohr^3	1.33	1.33315647	182.05	1.33			1.04		
$\varepsilon_{\text{HOMO}}$	eV	0.338	0.33752462	176.28	0.338			0.208		
$\varepsilon_{\text{LUMO}}$	eV	0.632	0.63145903	185.24	0.631			0.302		
$\Delta\varepsilon$	eV	0.723	0.72278386	179.67	0.722			0.373		
$\langle R^2 \rangle$	Bohr^2		55.51211190	175.93						
ZPVE	eV	0.0265	0.02646970	174.21	0.0265			0.0199		
U_0	eV	0.911	0.91071408	173.80	0.911			0.431		
U	eV		0.91190048	176.49						
H	eV		0.91197207	183.16						
G	eV		0.90933525	180.89						
C_v	$\text{kCal} \cdot \text{Mol}^{-1}$	0.907	0.90623131	187.81	0.906			0.777		
ω_1	cm^{-1}	131	131.43647496	185.36	131			13.2		

section, as a computational physics programmer, I think I will hesitate to use ML instead of DFT calculations. DFT is obviously a method recognized by the academic community. Although it is computationally intensive, the accuracy of the model is not necessarily better than the ML in the fitting interval. The reason why DFT is recognized by the academic community, I think it is because it has a clear physical meaning, there is Hohenberg-Kohn theorem; and this is not available in ML.

Go back to the topic. DFT itself has physical meaning, but if some DFT approximations give up most of the physical meaning to it, then we should not think that DFT is based on physics, but based on industrial applications. This is almost identical to the problem of ML solving chemistry, except that the model names of the two are different. We don't have to stick to the DFT approximation model and use the more efficient and efficient ML directly; just like a few beautiful vertical paintings and a few seconds of gorgeous animation can achieve or even detach from the ordinary movements, why not?

For a long time, ML couldn't be better than DFT approximation. After all, the error of the fitted set is there, plus the ML method to solve the chemical problem that should be able to use a strict physical basis, I am afraid it will not be accepted quickly in the short term; but this is probably temporary. The error of the fitting set will only become smaller and smaller, and the error of the ML model will only become smaller and smaller; in this case, the stubborn person will be shaken. So, are we going to continue to study DFT?

However, since I feel that although the DFT has undergone tremendous development thousands of years ago, future DFT theorists may become more and more lonely as the DFT approximation weakens. Even so, I think Hohenberg-Kohn will be very happy - I believe they didn't feel that this theory would be so vital at first. I also think that if we see precise functionals that they can't see, they must be scared in heaven.

[1] F. A. Faber *et al.*, J. Chem. Theory Comput **13**, 5255-5264 (2017).

[2] Ramakrishnan, Raghunathan; Dral, Pavlo; Dral, Pavlo O.; Rupp, Matthias; Anatole von Lilienfeld, O. (2017): Quantum chemistry structures and properties of 134 kilo molecules. figshare. Collection.

[3] J. C. Synder *et al.*, Phys. Rev. Lett. **1088**, 253002, 2012.

[4] J. Behlerp *et al.*, Phys. Rev. Lett. **98**, 146401, 2007.

[5] M. Rupp *et al.*, Phys. Rev. Lett. **108**, 058301, 2012.

[6] F. Brockherde *et al.*, Nature Communcationns. **8**, 872, 2017.