CS230

Machine Learning to Predict Electronic Structure Properties

Pranjal Agarwal[1] and Aisulu Aitbekova[2]
[1]Department of materials science and engineering, Stanford University, Stanford, CA, 94305
[2]Department of chemical engineering, Stanford University, Stanford, CA, 94305
pranjal9@stanford.edu
aait@stanford.edu

Abstract

ML is becoming ubiquitous in modern science. One appealing application of ML is to predict material properties such as electronic properties, which are traditionally determined using expensive quantum mechanical models. In this project, we explore various algorithms to predict spin-state splitting energy, a continuous variable, of transition metal complexes. We show that a neural network performs much better than linear regression and support vector regression models due to its ability to model non-linear relationships between input and output features. Additionally, we discuss ways on how to convert complex qualitative features into NN readable input using one hot encoder and a recursive binary tree.

Introduction

Electronic structure properties are material properties that could be described using interactions between electrons and nuclei in molecules. These properties can be determined using density functional theory (DFT), a computational quantum mechanical modelling method. The greatest advantage of DFT is high accuracy. The technique, however, is computationally expensive. This high cost drives the search of more efficient ways to predict the electronic properties. In particular, machine learning (ML) is becoming ubiquitous in modern science, where neural networks (NN) are trained using DFT generated data.

Here, we use the DFT data from Janet el al. (2017) to predict spin-state splitting energy, a continuous variable, of transition metal complexes.[1] The input to our algorithm is a set of quantitative and qualitative features describing the properties of various metal complexes. The data set was split into training, development, and test sets. To convert qualitative features into quantitative ones, we used one-hot encoder. In terms of a NN architecture, we implemented a NN with one hidden layer consisting of 100 units, '*Relu*' activation function, and '*Adam*' solver. We demonstrated the advantage of the NN compared to linear regression with and without regularization and support vector regression (SVR). The advantage of the NN stems from its ability to model non-linear dependency between the input features and the output variable.

Related work

ML for material property prediction has two characteristic features. First, material discovery problems have relatively small training data sets. The scarcity of the training data is explained by the need to perform experiments or DFT calculations. Experiments usually take long time, while

DFT becomes computationally expensive for large molecules and structures. Second, the data consists of complicated features such as 3D molecular structures and 2D formulas, thus, feature selection and data processing are often tricky. To convert the qualitative features into quantitative values that can be fed into a NN, researchers use several approaches. For example, Cooley Lab at MIT uses *Chemprop* software to convert molecular structures into NN readable input.[4] Other works utilize graph neural networks such as *MatErials Graph Network*, a publicly available implementation of DeepMind's graph networks.[5] Some examples on the successful application of NN to predict material properties include works by Janet et al. (2017), Gomez Bombarelli et al (2016), and Pyzer-Knapp (2015).[1,6-7] The advantage of NN is that it allows to model non-linear relationships between input and output variables. Because of relatively small size of employed data sets, feature selection plays an important role.[1] Feature selection, traditionally performed manually, can now be implemented using recursive trees that allow to understand which specific features contribute the most to the output variables.[1] LASSO regression also finds a wide application in the feature selection field because it shrinks original features into a smaller set of the most important ones.[1] The disadvantage of this approach, however, is potential loss of important structural data.

Dataset and Features

Our data set consists of 1345 examples, where each example has 17 features describing various chemical and physical properties of metal complexes. Since 5 out of 17 features included qualitative information such as an element type, symmetry group, and types of chemical bonds, we converted these features into quantitative ones using one-hot encoder. Thus, we obtained 58 features per example. The data set was shuffled, normalized, and split into training, development, and test sets with the relative ratios of 60:20:20, respectively.

Methods

As a starting point, we used multivariate linear regression to demonstrate that the linear model cannot capture non-linear relationships between the input and output. The cost function was defined as following:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)^2$$

where θ is a parameter vector in R^n, where n is the number of features, $h_\theta(x^i)$ is a hypothesis value of example i, m is the total number of examples, and y^i is the output of example i. The objective of the regression model is to minimize the cost value J. The linear regression problem has a global minimum. Optimal model parameters can be solved using gradient descent, where with each iteration the parameter vector θ is modified in the following way:

$$\theta_j = \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_j^i \text{ , for j = 1,2,3,…,n}$$

Here, α is a learning rate. If the learning rate is too small, the algorithm learns slowly. If the rate is too high, the cost diverges with the number of iterations. The optimal learning rate is chosen by plotting cost versus number of iterations for various values of the learning rate and picking the one that performs the best on a cross-validation set. Alternatively, θ can be computed directly using

normal equations, but this approach involves inverting X^TX, and this operation gets slow as n becomes large.

The linear regression model suffered from both high bias and high variance. To reduce the variance, we introduced a regularization parameter λ and modified the cost function as following:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)^2 + \lambda\sum_{j=1}^{n}\theta_j^2$$

By tuning the value of λ we can control the extent of regularization: the larger the λ, the larger the regularization. Usually the optimal value of λ is chosen by running the algorithm using various values of regularization parameter and picking the one that gives the best performance on a cross-validation set.

Although the inclusion of the regularization parameter reduced the cost on the test set, the model still suffered from high bias because the linear model was unable to capture non-linearity of the problem. Therefore, we a NN consisting of one hidden layer with 100 units, 'Relu' activation function, and 'Adam' solver. Despite its simple architecture, the NN significantly improved the prediction accuracy on the test set.

Results and Discussion

Table 1 and Figure 1 summarize the results from the three models and the original work.

Table 1. Cost values

Model	Training	Development	Test
Linear regression	89.39		2.45e24
Linear regression with regularization (λ = 0.001)	88.42	103.32	96.86
MLP regressor	2.55		11.12

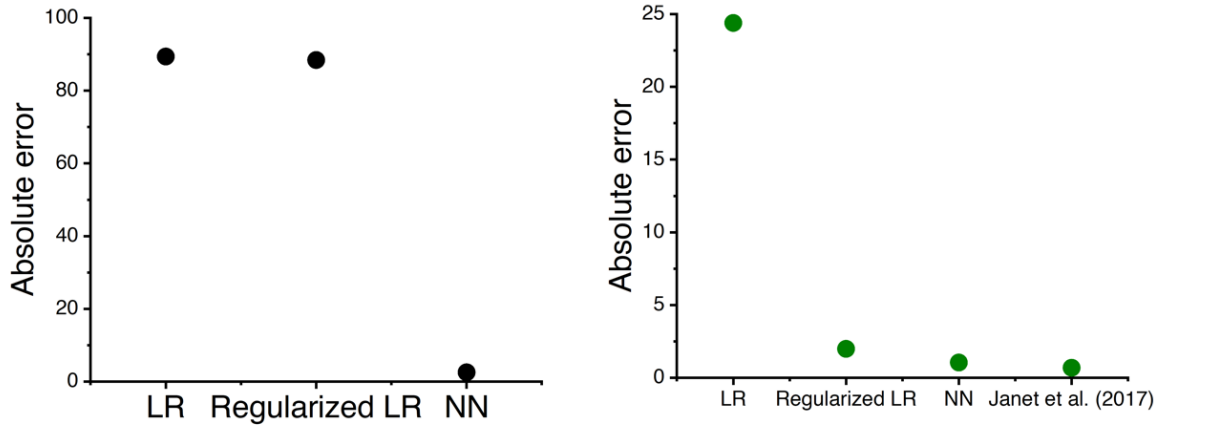

Figure 1. Absolute error on the training set (left) and test set (right) for linear regression (LR), linear regression with regularization parameter, and neural network (NN).

The primary metrics is the cost function, which is computed as a difference between predicted actual output values. Table 1 shows that the linear regression model did not give the satisfactory results. The cost on the training set was 89.39, but was on the order of 10^{24} for the test set. The model suffered from both high bias due to the inability of the linear model to capture non-linearities between the input features and the output variable and high variance due to overfitting. To decrease the variance, we included regularization. To find an optimal regularization parameter lambda, we ran linear regression using multiple lambda values from 0.001 to 100 and used the cross-validation set to pick the optimal lambda value. Adding regularization helped to reduce the cost values on both the training and test sets. Strangely, we observed that the cost on the training and cross-validation sets are independent of lambda until it reaches 100 (Figure S1). To solve the high bias problem, we implemented a NN with one hidden layer consisting of 100 units, '*Relu*' activation function, and '*Adam*' solver. Despite its simple architecture, the NN gave the smallest cost.

Realizing the shortcomings of the one hot encoder, we wanted to devise a better way to convert the qualitative features into quantitative. We, thus, implemented recursive binary tree following the previously published procedures.[8] Although we were not able to feed the data obtained from the binary tree into the NN, we identified which features had the strongest contribution to the output variable. Concretely, we determined that the type of ligands (Cl, O, S elements) have the greatest influence on the spin-state splitting energy.

Conclusion and Future work

In conclusion, NN is the most appropriate choice given the non-linear relationships between the input and output. In terms of future directions, it would be desirable to improve the accuracy of the proposed model and find a way to incorporate the results from the recursive tree into the NN.

Contributions

Pranjal Agarwal and Aisulu Aitbekova equally contributed to this work at all stages of the project.

References

1. J.P. Janet and H.J. Kulik 2017, *Chem. Sci.* 8, 5137
2. L. Breiman , J. Friedman , R. A. Olshen and C. Stone , *Classification and Regression Trees* , Chapman and Hall, CRC, 1984, vol. vol. 5, pp. 95–96
3. T. Therneau, B. Atkinson, B. Ripley, Rpart: Recursive Partitioning and Regression Trees, https://cran.r-project.org/package=rpart
4. Chemprop v0.1 http://chemprop.csail.mit.edu
5. Chi Chen, Saaketh Desai, Yunxing Zuo (2021), "Materials Graph Network," https://nanohub.org/resources/megnet. (DOI: 10.21981/WYQB-XD49).
6. R. Gomez-Bombarelli, J. Aguilera-Iparraguirre, T. D. Hirzel, D. Duvenaud, D. Maclaurin, M. A. Blood-Forsythe, H. S. Chae, M. Einzinger, D. G. Ha, T. Wu, G. Markopoulos, S. Jeon, H. Kang, H. Miyazaki, M. Numata, S. Kim, W. Huang, S. I. Hong, M. Baldo, R. P. Adams and A. Aspuru-Guzik 2016, *Nat. Mater.* 15, 1120–1127.
7. E. O. Pyzer-Knapp, K. Li and A. Aspuru-Guzik 2015, *Adv. Funct. Mater.* 25, 6495–6502.

8. L. Breiman , J. Friedman , R. A. Olshen and C. Stone , *Classification and Regression Trees* , Chapman and Hall, CRC, 1984, vol. vol. 5, pp. 95–96

Appendix

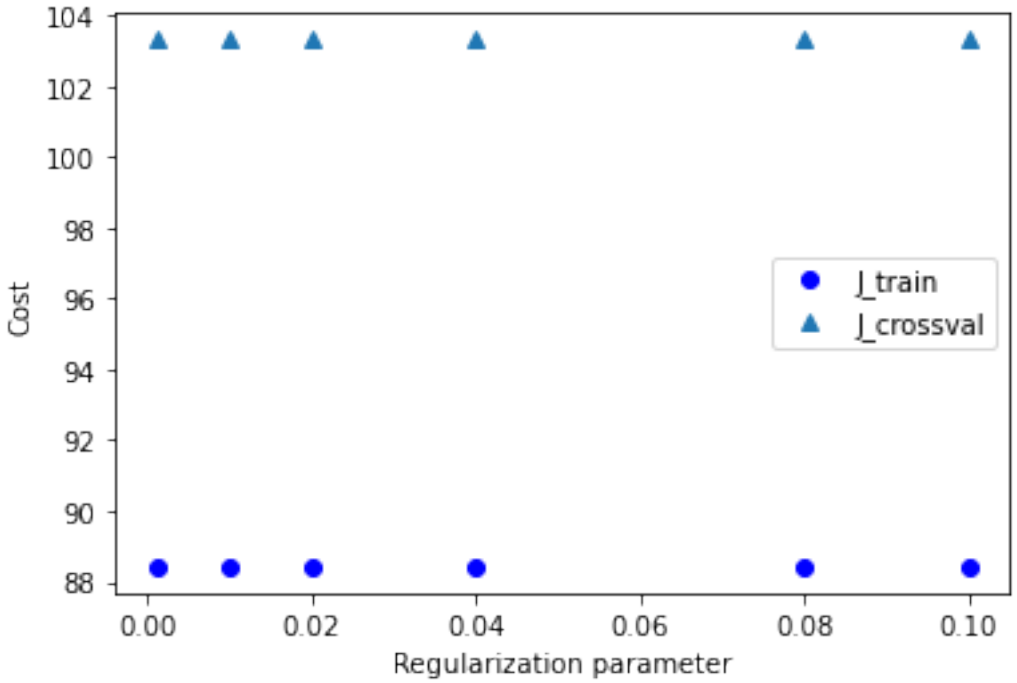

Figure 1. Training and cross-validation costs versus regularization parameter.