CS 181 – Practical 1:
Predicting the Efficiency of Organic Photovoltaics

Group 20: Bolei Deng, Matheus C. Fernandes

February 10, 2017

Introduction

For this practical, we were given unidentified data on features of chemicals that produce a given energy difference between the highest occupied molecular orbit and the lowest unoccupied molecular orbit. This difference, also referred to as the gap, identifies how efficiently the material can serve as an organic photovoltaic for clean energy applications. We approach this project by using various Machine Learning (ML) algorithms to learn from the training data and predict what the gaps are. In theory, once we have efficiently incorporated a learning algorithm, we can use different optimization methods to obtain the best organic compound that optimizes the potential gap (though, this part it outside the scope of this practical assignment.)

1 Technical Approach

Data Preprocessing

For our approach, we fully focused on the built-in functions found in *Scikit Learn*. Although we ran across multiple other ML libraries, we decided to focus on *Scikit Learn* as it contained the widest array of tools we needed to complete this practical. We began our analysis by focusing primarily on how we can most efficiently use the data provided. Our first step was to check how many of the features do not contribute to the learning algorithm. Given that the features are all in binary form, we can conclude that if the average of a feature for all samples are either 1 or 0, they are not contributing much in terms of learning to the algorithm. In other words, we have taken the mean of all the values for each feature such that $\mu_f = \frac{1}{N}\sum_{i=1}^{N} x_i$, where N is the total number of samples, x_i is the i_{th} value for a given feature, and μ_f is the mean for each feature. So if $\mu_f = 0$ or $\mu_f = 1$ we discard the feature column. Furthermore, we also look for values of μ_f that are very close to 0 or 1 ($\sim 10^{-4}$ or $\sim 1 - 10^{-4}$). This means that there is a small number of values that differ from from 0 or 1, respectively for a given feature through all samples.

Training & Model Experimentation

As predefined by the problem statement, we quantify the accuracy of our methodology by considering the *Root Mean Squared Error* (RMSE). Although, many of the methods that we explore do not use RMSE as their loss function, we must consider how our model performs under the RMSE measure. Furthermore, given the limited amount of training data provided, we must infer how well each model performs for the given features. To quantitatively analyze this, we consider the cross-validation

method in which we split the data set provided into 5 equally distributed randomly selected sections ($5 \times 20\%$ sections). Out of these sections, we keep 1 as a the test section and 4 as the training sections. This ensures that we are using the most appropriate model as possible before touching the test data provided. In general, we train the model by using the training data and then obtain the RMSE of the training data by feeding it back into model. We then obtain the RMSE of the training data by comparing it to the known values. We do the same thing to the test set, by comparing the values obtained from the model to the known values of the test set through RMSE. We then compare the RMSE of the training set to the test set and we can infer whether or not we are over-fitting the model to the training data. When running ridge regression, lasso, or elastic-net, we used the difference between the training set error and the testing set error to tune the regularization constant, as a large difference between these two values is indicative of over-fitting.

Furthermore, through linear regression, we have found that the gap not only depends on the features themselves, but also on the nonlinear FOIL product terms of each other. For this, we only take the 1st order cross-dependent terms and not the squared terms. Such that the new features become

$$\boldsymbol{f}_2^{ij} = \boldsymbol{f}_i \times \boldsymbol{f}_j \quad \text{for} \quad i \neq j \quad \text{where} \quad i, j \subset [1, M],$$

and $\boldsymbol{f}_i$ is the ith feature, $\boldsymbol{f}_2^{ij}$ are the additional features after the single term features, and M is the total number of features (not including the new ones.) We also investigate how a third term impacts the regression, such that,

$$\boldsymbol{f}_3^{ijk} = \boldsymbol{f}_i \times \boldsymbol{f}_j \times \boldsymbol{f}_k \quad \text{for} \quad i \neq j \neq k \quad \text{where} \quad i, j, k \subset [1, M]$$

and $\boldsymbol{f}_3^{ijk}$ are the additional features after the single term features. The reason we do not consider the squared and cubic terms in the above two cases is that they do not introduce any new features (because they are in binary, meaning $1^2 = 1$ or $0^2 = 0$.)

2 Results

As discussed in previous part, we improve the linear regression model by taking some new features, i.e., cross-dependent terms, into consideration. We start by assuming that only the adjacent terms interacts with each other. Specifically, in addition to the 28 basic features $\{\boldsymbol{f}_i\}$, we add another 27 features, which is $\{\boldsymbol{f}_i \times \boldsymbol{f}_{i+1}\}$. Furthermore, since we do not understand the underlying chemical property of the smiles, features apart from each other may also interacts with themselves. To consider this, we keep adding features $\{\boldsymbol{f}_i \times \boldsymbol{f}_{i+p}\}$ to the model. Here we call p the "maximum distance" (depending on the order of the features), and we assume that there is no interaction between features with distance beyond p. Note that $p = M - 1$ means we have considered all possible feature pairs in the model. Moreover, as indicated in Section 1, we also investigate the three feature interactions $\{\boldsymbol{f}_3^{ijk}\}$ in addition to the two feature ones. Similarly, maximum distance p means only feature interactions within distance p are considered. In fig. 1 (a), we illustrate the RMSE of predictions with different maximum distances, i.e., $\{\boldsymbol{f}_2^{ij} | (j - i) < p\}$ and $\{\boldsymbol{f}_3^{ijk} | (k - i) < p\}$. In fig. 1 (b), we present the number of features in linear regression with different maximum distance p.

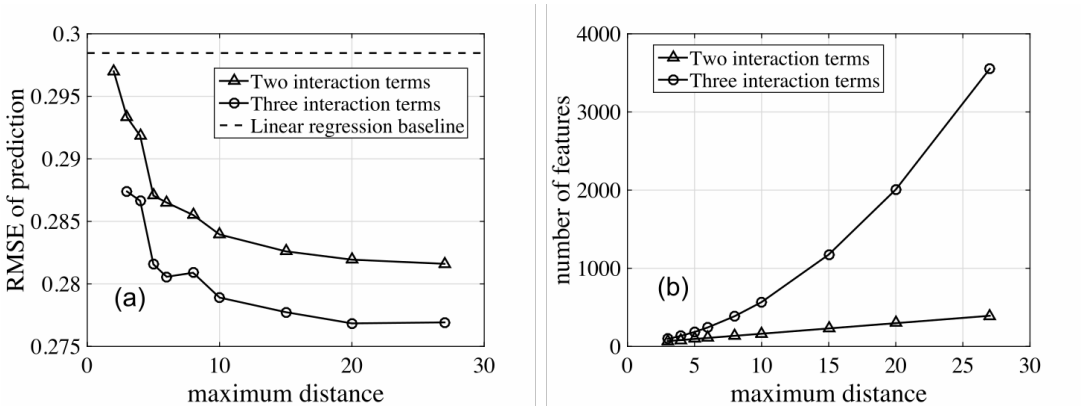

Figure 1: (a) the RMSE of predictions for different maximum distances considering two interaction terms and three interaction terms respectively. The dash line on the top indicating the original linear regression baseline; (b) number of features when different maximum distances are evaluated.

3 Discussion

As illustrated by fig. 1 (a), by introducing these new interaction (cross-terms) features, we are able to produce better performance comparing to the original linear regression (presented by dashed line). As we increase the maximum distance, the predictions becomes more and more accurate (fig. 1 (a)), which is because we are considering more features within our model (fig. 1 (b)). We also notice that the performance of our algorithm saturates when $p > 10$, although the number of features keeps growing. This means that interactions between features decay as you increase the distance between themselves. This proves our assumption that the gap also depends on the nonlinear FOIL product terms of each other. In another words, the influence of the coexistence of two features is not equivalent to their linear summation. To further improve the model, one could consider adding more features by including higher order FOIL product terms. This would lead to terms that are dependent on four different features at the same time.

We have also considered multiple other algorithms including: Random Forest (including Decision Trees by themselves), Neural Networks, Elastic Net, MLP Classifier, MLP Regressor, and many others. Through trial and error, we have found that a lot of these tools have not improved the predictions because of the data's binary nature. While working with the Decision Trees, we were able to get the best model with depth set to 44, which is around the number of features we are considering once we purge the useless through the data.

As can be seen from the previous section, our model does a relatively well job on fitting the data and predicting the gap based on the features provided by the test data. However, the model training and ultimately the predictions could be further improved by considering more features for each composition. Such feature engineering strategy could be achieved by using the provided *smiles* in combination with the *RDKit* package to generate additional features for the model. We decided not to pursue this approach, and instead we focused on using the data that was provided to us. As a challenge, this asserts that we can develop/tune a model which is able to best predict the gap based on the fewest amount of information.

Practicals on Github: fer.me/cs181-practicals