Practical 1: Regression
Predicting the Efficiency of Organic Photovoltaics

Antonio Copete (acopete@cfa.harvard.edu, Camelot.ai Copete)
Fangli Geng (fgeng@g.harvard.edu, Camelot.ai fangligeng)
Xihan Zhang (xihanzhang@hsph.harvard.edu, Camelot.ai Xihan)

February 10, 2018

1 Technical Approach

We began by performing exploratory analysis on the sample dataset we were given, which consisted of a training set of 1 million molecules with 256 binary predictors, in addition to their SMILES string and the HOMO–LUMO gap we were seeking to predict. The test set consisted of 824,230 molecules with the same set of predictors as well as their SMILES string. The sample code we were given implemented a default Linear Regression model on the full set of 256 predictors, yielding an $R^2_{\rm LR} = 0.461 (\text{RMSE}_{\rm LR} = 0.298)$ over the full training set, and a default Random Forest regression with $R^2_{\rm RF} = 0.554 (\text{RMSE}_{\rm RF} = 0.272)$.

Initial inspection found that out of 256 molecular features, 221 of them were unexpressed (i.e. had $x_i = 0$) for *all* molecules, both in the training set and the test set. Dropping unimportant features would normally call for K-fold cross-validation across the training set to ensure those features are consistently unimportant in all cases. However, in the case of null values of certain features for every element of the training set, it is not only legitimate but necessary to drop those features from all further analysis, as fitting along null dimensions would constitute a form of overfitting.

Having reduced the sample dataset to 31 expressed molecular features, we performed both regularized and non-regularized linear regression with cross-validation[1] and hyperparameter tuning, under the following methods:

1. Non-regularized linear regression: Yields a mean $R^2 = 0.461$ from 5-fold cross-validation, similar to the original result with 256 predictors.
2. Ridge Regression: Tuning the hyperparamerter α by grid-search cross-validation results in an optimal value of 5.86×10^{-5}, with a best $R^2 = 0.459$, a result almost identical to non-regularized regression.
3. Elastic Net: Trying a more general regularization of the form $\alpha_1 L_1(\text{Lasso}) + \alpha_2 L_2$ (Ridge regression), and tuning the $L_1\text{ratio} \equiv \alpha_1/\alpha_2$ by grid-search cross validation, yields an optimal

[1] After initially trying 3-fold, 5-fold and 10-fold cross-validation, we settled on 5-fold cross-validation as the best compromise between accuracy and computational speed.

value of 0.0, which indicates that that a regularization biased towards sparse models, such as the Lasso, might not be the best choice in this case.

The results from linear regression on the sample dataset consistently gave us relatively low predictive value, even when compared to generic random forest regression, which led us into 2 parallel tracks:

1. Feature engineering:

   The RDKit package in Python use the SMILES string to provide an array of information and functions about a molecule, including substructure searching, chemical reaction, enumeration of molecular resonance structures, and several sorts of fingerprints. From our research we decided to extract the *Morgan fingerprint function*, which is widely used with SKLearn in ML and carries the information we considered to be most relevant to the efficiency of organic photovoltaics, such as: (a) Connectivity: Element, numbers of heavy neighbors, numbers of Hs, charge, isotope, inRing; (b) Chemical features: Donor, Acceptor, Aromatic, Halogen, Basic, Acidic; and (c) Taking into account the neighborhood of each atom.

   The process takes the "SMILES" column from the training and test data and uses RDKit to convert it into molecule objects, followed by gathering of the Morgan fingerprint by use of the command `rdkit.Chem.getmorganfingerprintasbitvect()`, adjusting the parameter `nBits` to 512, 2048 and 4096 aiming to test datasets with various precision levels. After iterating this process for each string, we appended the columns together to form a new input matrix. The size of the dataset greatly varies with precision level, and for this reason we had to slice the datasets into 8 parts in order to accommodate computational constraints.

2. Non-linear methods:

   Among these we concentrated on 2 broad categories:

   (a) Tree-based Ensemble methods:

   Given the binary predictors coincide with the branch-like structure of decision tree, we had a deeper dive into decision tree. Deep-depth tree may overfit the training set, so that ensemble methods were applied to take the average of predictions of all trees. The baseline ensemble methods are: (a)Bagging : boostrap multiple training sets to fit multiple trees to reduce the variance.(b)Randomforest : further reduce the variance by randomly picking sub set of $\sqrt{p}$ predictors from the total p predictors at each split to make.(c)Extremely Randomized Trees : futher reduce the variance by selecting cutting point from sub set totaly at random.(d)AdaBoost: use a linear combination of multiple simple trees to reduce the residual step by step.

   Baseline models were fitted on the selected 31 features. 20% validation set was used to test MSE generated by each model. Bagging, Randomforest, Extremely Randomized Trees are quit similar to each other. Adaboost which focus on reducing bias rather than varriance, is much weaker. Thus, we selected random forest as main model, and tuned number of estimators(i.e. the total number of trees to take average on) to reduce variance of the model. 5-fold cross validation was used on the 1/7 of the total sample with 2048 newly generated features.

From result, 13 estimators is sufficient to make MSE of test set near training set. Given the total sample size is 7 times of the test set where we tuned the model, setting the number of estimators to be 150 for the whole data set is safe and reasonable.

We also plot the distribution of importance of predictors based on the baseline random forest, the result is as below:

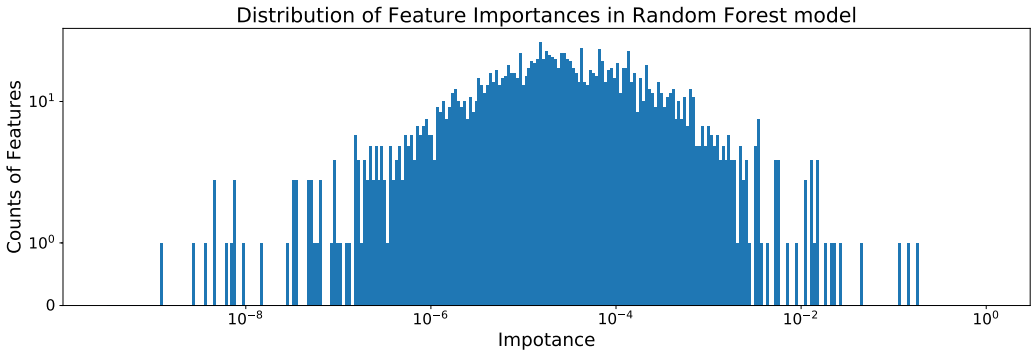

Figure 1: Histogram of feature importances from Random Forest

Most portion of predictors have little importance(10^-8 to 10^-2), but combining them can yield good precision on validtion set($R^2 = 0.936$, MSE=0.0106). This made us explore more on Deep learning method, which considers the interaction among features.

(b) Deep learning:

Since the predictors describe a broad set of molecular properties, it is natural to expect non-linear interactions between them to play a role in determining the HOMO–LUMO gap. For this reason we decided to explore the training of simple neural networks that would serve to model these interactions and got acquainted with the basics of Deep Learning as a tool to tackle this problem. In all cases, we used an architecture of one or more dense (fully-connected) hidden layers between the input layer of N parameters and the output layer of 1 node for the response we wanted to model. For each node in the hidden layers, we used the widely used "relu" function ($f(x \leq 0) = 0, f(x > 0) = x$) as activation function. We trained the model using the *keras* module in Python, using the *adam* optimizer for the learning rate, and setting aside 20% of training data for validation. The model would be fitting for the weights of the connections between pairs of nodes in the network, which in the case of 1 hidden layer with M nodes, would result in $(N + 1) \times M$ parameters to be trained.

Taking a trial-and-error approach to setting the network architecture, we show the results of validation set MSE for 3 broad cases, using the 31 expressed features in the sample training dataset: 1) 1 hidden layer with variable number of nodes, 2) 2 hidden layers, with 31 nodes on the first and variable number of nodes on the second, and 3) 3 hidden layers, with 31 nodes on each of the first two, and variable number of nodes on the third.

The result of this experiment showed that the MSE was not as sensitive to the addition of a second and third layer, as it was to the number of nodes in the first layer of the

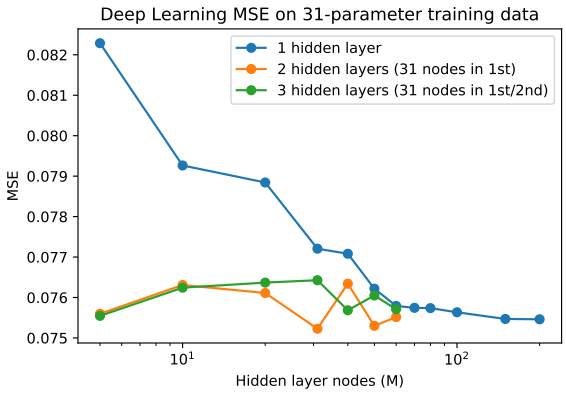

Figure 2: Experimentation on DL Neural Network architecture

1-hidden layer architecture. Given the computational limitations that were expected for a much larger set of features, we used this result to use the feature-engineered set to train only 1-layer models with a limited number of nodes in the same range as the number of predictors.

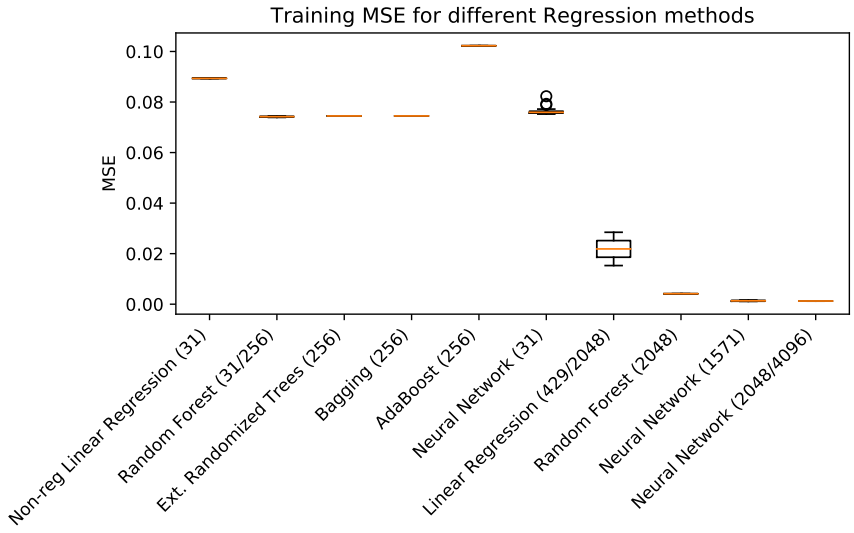

Figure 3: Overall results for different regression methods

2 Results

Table 1 displays the performance of models we tested after we got the new feature information from RDKit.

1. Feature set selection

To select best feature set, we test the performance of the model we have experimented on as mentioned in Technical Approach:

(a) Linear Regression

We first used 2048 features (which is default in RDKit function) and tested the linear regression. Its results gave us a better result (RMSE 0.1236) than that the best deep learning model (Antonio please put a number here) gave us with 256 features. It indicates the big loss of information using 256 features.
For the purpose of further exploration and reduction of computational burden, we tried to drop the features which have coefficient = 0 from linear regression model results, however, only 1% of the 2048 features can be dropped. We increased the threshold of dropping data until abs(coefficient)¡=0.15, this finally gave a subset with 429 features. We then ran linear regression again with the new subset, but the final RMSE was increased by 30%.

(b) Random Forest

We also ran random forest model with 512 features and 2048 features, but it showed that the 2048 features result was much better than the others.

These two sets of comparison demonstrated that either 429 features selected by linear regression coefficient or 512 features directly given by RDKit fingerprint function was not informative enough. Therefore, we decided to test tune different models with 2048 features, which give us a relatively good amount of information for the molecules energy efficiency prediction.

2. Tuning on neural network

As discussed in the technical approach part, the experiments using original dataset (256b features) show that neural network with one layer could provide the most accurate results. So we decided to test neural network model With 2048 features.

We set up 3 different configurations to conduct the experiments with neural network:

(a) 2 layer neural network with 2048 features and 100 units in each layer;

(b) 1 layer neural network with 2048 features with 1000 units;

(c) 1 layer neural network with 4096 features with 1024 units.

Comparing the results between 1st and 2nd experiment, we found the layer number is not as important as unit number, which is consistent with what we found using the smaller number of features. Comparing the result between 2nd and 3rd experiment, we found that 2048 Morgan fingerprint is basically informative enough to predict, and 4096 features in more detail couldn't improve the prediction as much but meanwhile taking a much larger amount of time for computation.

As 4096 features with 1 layers and 1024 nodes didn't provide the prediction as precise as we expected, we were turning to find additional features and found counted-based fingerprints could be a good option in addition to the bit-based fingerprint we used. Count-based fingerprint count the number of times a feature appears instead of simply that it appears, which

could potentially provided new and important information to our prediction.

3. Other methods not realized

   We also wanted to try other algorithms like Principal Component Analysis to select more important features to reduce computational complexity, and to implement neural network models with 2 layer and 2000 nodes to allow more complex interactions between selected important features. However, due to the time constraints we had, we didn't realize those plans.

4. Summary

   As a summary, we found that the linear regression and random forest are good methods to rapidly select useful features. Neural network is much more accurate in non-linear regression, especially with large dataset and rich features condition.

5. Box Plots for the performance of other tested models

   (Antonio): Main result here should be a box plot with the MSE of different methods we tried. Models for the original data (256 features) should be different from models for the dataset of all the features.

3 Discussion

We performed exploratory analysis on the 256 features by fitting Linear Regression and Random forest as baseline models. For feature space, after filtering features with all 0 input, only 31 features left, which gave too little information compared to the 100000 data points. Thus, we use RDkit to generate 2048 features. For model selection, we gave deeper dive into 2 tracks of non-linear models: Tree-based ensemble methods and deep learning, based on the fact: no matter tunning on linear methods the performance can't beat non-tuned random forest.

Given the branch-like input nature, we first tried tree-based ensemble methods, which much more improve the performance on prediction. However, the fact that most portion of features are of little importance and the fact that each key in chemicals has high interaction with others, drove us to focus more on Deep learning method, which considers the interaction among features. The deep learning method did significantly improve the performance on prediction.

One the other hand, the newly generated 2048 features also largely improved the precision of any kind of methods we experimented on 256 features. Thus our final prediction was made by deep learning method based on the 2048 features.