Practical 1: Regression

William Burke, Buse Aktas, Matthew Holman
wburke@g.harvard.edu, buseaktas@g.harvard.edu, mholman@cfa.harvard.edu
wburke, BuseAktas

February 9, 2018

1 Technical Approach

1.1 Introduction: goals and scoring criteria

Organic photovoltaic molecules could be used to manufacture cheap, flexible solar cells. To determine these molecules' potential efficiency in solar cells, the Harvard Clean Energy Project has been using density functional theory. The energy gap between the highest occupied molecular orbital (HOMO) and the lowest unoccupied molecular orbital (LUMO) is the main quantity of interest for these purposes; in this assignment we sough to predict the HOMO-LUMO energy gap for unknown molecules. We experimented with several classes of models, training them on a given dataset of 1000000 molecules and gaps, and then attempting to predict the gaps for 824230 previously unknown molecules. We scored each of the models using root mean squared errors (lower is better). If there are N test data, where your prediction is $\hat{x}_n$ and the truth is x_n, then the RMSE is

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(\hat{x}_n - x_n)^2}$$

We scored our models using several datasets. In our local environments, we scored our models against various validation sets created from a subset of the given training data. We experimented with different approaches to generate validation sets. In some trials, we withheld an arbitrary number of points (i.e. the last 10% of the data) from our training set, using the majority of the data to train our models, and the validation data to evaluate their performance. In other cases, we selected the validation data points randomly from the training data, and followed the same procedure. Our validation results were mostly stable across models and validation strategies. After selecting the best models using our hypothesis about the relative merits of each model and the training times and results from our local tests, we generated predictions for the test data and uploaded the predictions to Camelot. Although we still do not know how well our models performed on the test data, Camelot used a subset of the test data to generate a public leaderboard and RMSE scores. These are the Test RMSE scores reported in Table 3.

1.2 Features: tuning and configuring data

We have one million molecules in our training set, along with floating point values for the HOMO-LUMO gap. The test set is in an identical format, except that no gap data was provided. For each

Summary of features used

Feature	Count
Original Features	256
Original Features, Cleaned	31
Morgan Fingerprint Features	1024
Morgan Fingerprint Features, Cleaned	1003

Table 1: Cleaning out features that were all 0s or all 1s significantly reduced the size of the original dataset, indicating it was very low quality. By contrast, very few of the Morgan fingerprint features were found to be all 0s or all 1s.

molecule we have the SMILES string, which is a unique representation of the molecule. Here is an example of a smiles string:

c1sc(-c2sc(-c3ccc(cc3)-c3scc4sccc34)c3[se]ccc23)c2sccc12

In addition to our SMILES strings, we were also given a 256-dimensional binary array corresponding to each of these molecules. Presumably, these were extracted using the Python package RDKit, but we were not given any further information on how these arrays were generated. We also have opportunity to use the RDKit package ourselves to be able to extract our own set of features with different dimensions.

In our first experiments, we set up a framework to train 8 different types of models against our data. Our code was designed to output a validation score for each model. We quickly realized that many of the unknown features were useless for prediction purposes - in the combined training and test sets, there were many features which were always 0 or 1, providing no useful information about variations in the gap values. We modified our code to remove any feature which took a constant value. Incidentally, we included a check to see if there were any features which varied in the test set but not the training set, and vice versa, but did not find any such features. As shown in Table 1, this cleaning procedure reduced the number of features from 256 to 31.

Although we do not have visibility into the creation process for the initial 256 features, the number of useless features we discovered made us skeptical of the quality of the data. Our validation results are reported in Table 2, but as we confirmed by submitting predictions for the test data to Camelot, the data was low quality. None of our 8 initial models did better than an RMSE of 0.27. This was enough to beat the benchmarks, but we immediately decided we needed better data.

Our next challenge was generating new features using RDKIT. After 30 hours of unsuccessful attempts, 2 visits to office hours, and extensive help from Piazza and the course staff, we finally managed to install RDKIT on one of our machines. However, this was not the end of our struggle. We made several attempts to generate vectors of features and pass them to our code as pandas DataFrame objects, to no avail. Ultimately, we split the computational work between the computers of the teammates and borrowed our friend's computers for additional horsepower. After 6 hours of "parallel processing," we finally succeeded in creating test and training data sets of 1024 Morgan Fingerprints. Morgan Fingerprints are a bit-vector representations of molecular structure that can be readily used as features in model fitting. Applying our cleaning procedure reduced the size of the data set to 1003 features. Once again, we included a check to see if there were any features which varied in the test set but not the training set.

We considered combining the Morgan Fingerprints with the small, low quality data set, but decided to test them separately for several reasons: (1) The small data set was low quality; it was not clear if we would just be adding irrelevant features to the model. (2) By increasing the number of features,

Model	Validation RMSE.	Test RMSE.
LinearRegression	0.0905	
RandomForestRegressor	0.0754	0.27208
LassoCV	0.0906	0.29850
RidgeCV	0.0905	0.29845
ElasticNet	0.1663	
MLPRegressor	0.0777	0.27496
BaggingRegressor	0.0754	0.27208
GradientBoostingRegressor	0.0828	0.28592

Table 2: We tried most of the suggested models on our cleaned version of the original data set. In this table, we report the initial root mean squared errors that we found when testing the trained models against our the validation set.

we would introduce the possibility of overfitting. (3) With limited computational resources, it made sense to focus on our highest quality data. (4) We had no idea how the small data set was generated; Moran Fingerprints are well documented and there are plenty of resources online explaining how they are generated.(5) Our small data set might include duplicates of the features in our large data set.

1.3 Models: exploration of different methods and deep dive into bagging

Our next challenge was training our models. Given the vastly increased size of our data set, we knew it would take a long time to train each model, and even with our "parallel processing" approach, we realized that we could try perhaps three models before Camelot closed. To decide which models to train, we had an intense discussion about the pros and cons of each approach and leaned heavily on what we had learned from our tests with the small data set. Ultimately, we decided to train a linear regression model as a benchmark and compare it to two ensemble methods, bagging and boosting.

Bagging and boosting are both ensemble methods that combine the results of a number of base regressors (or classifiers) that use a subset of the original data set, sampled with replacement. Boosting adjusts the weights of the different base classifiers to concentrate on the difficult, poorly predicted or fit cases.

Using the linear regression model as a baseline, we compared other methods of regression to test and gauge how other methods performed. We saw that the bagging method was unsurprisingly the most efficient one. Bagging for regression takes random samples from the training set, fits an individual model for each set, and then takes the average of the predictions to predict the new test data set. This method, since it takes the average of all the models, tends to decrease the variance without dramatically changing the bias. Penalizing overfitting without oversimplifying the model helps us have a model which performs better on making predictions for a test data set.

Also, given that we did not have a clear scientific insight on how the HOMO-LUMO gap might relate to each of the different features we extracted from the fingerprints, it was crucial that these methods weighed the importance of each of these features numerically. We might have considered features which don't have an effect on the gap. The methods of regression helps us diminish the impact of the independent features to the model. Although this helps, it doesn't take the place of a more in depth understanding of the phenomenon studied which would have allowed us to make

Model	Validation RMSE.	Test RMSE.
LinearRegression	0.02050	0.14337
BaggingRegressor	0.00085	0.06456
GradientBoostingRegressor	0.03098	0.17571

Table 3: We trained three of the suggested models on our cleaned version of the Morgan data set. In the first data column of this table, we report the initial root mean squared errors that we found when testing the trained models against our validation set. In the second column, we report the final root mean squared errors that we found when testing the final trained models against the test set.

smarter decisions regarding the dependencies. There might have been a bias in the data, such that molecules in the training data set with large HOMO-LUMO gaps also have large values for certain features. The fact that these values correlate doesn't necessarily mean causation. Having a large data set of one million molecules hopefully diminished the possibility of such occurrences, yet since we also do not have a particular idea of how the training data was collected, it is also difficult to judge.

2 Results

We have made predictions using different models, as well as different forms of training data:

- We started forming our predictions, first by using the original cleaned features as our training data set, which only had 31 features out of the 256 given. We separated this original data into a training and a validation data set, and used the validation set to check and compare the RMSE values for the different models. These values can be seen in Table 2.
- We then applied the models we had trained and validated to the test data set and compared the RMSE of that. The RMSE values were higher when predicting the data set compared to predicting the validation data, yet the ranking of how effective each different model was remained effectively the same.
- Finally, we chose to move forward with the two ensemble models which we had found to be the most effective, alongside a simple linear regression for benchmarking purposes. Since we had realized that the feature engineering done in the original dataset wasn't giving satisfactory results, we extracted all the possible 1024 features from the Morgan Fingerprint of each molecules. We applied the same cleaning procedure to avoid any extra computational work. Using these larger set of features as our training data we achieved much better RMSE results for our test data, showing the importance of feature engineering. The results for these predictions can be seen in Table 3. We submitted these results to Camelot.

3 Discussion

As described an earlier section, our approach to this problem was to first work with the set of features that were provided and to use the two methods in the sample code, just to get started. The initial goal was just to make sure that we understood the problem, the data, and simple solutions.

Next, we used nearly every regression method that was readily available in order to try to improve the performance without having to invent new methods or to develop new features. This gave a sense of which methods appeared to perform the best with the original feature set.

Finally, we used RDKit to extract a larger and better posed set of features. Given the limitations of time, for this next step we used just those few regression methods that performed the best on the original feature set, under the assumption that these same methods would also be the best with an extended feature set. We could have tested that assumption given more time.

The adaptations we made both through model selection and through feature engineering clearly improved our results as it can be seen from the reported values of the RMSE.