Practical 1: Predicting the Efficiency of Organic Photovoltaics Repository: https://github.com/tongbaojia/pubcs181

Baojia Tong, baojia.tong@cern.ch
Yuliya Dovzhenko, dovzhenko@g.harvard.edu
Alan Le Goallec, alanlegoallec@g.harvard.edu

February 10, 2017

We were asked to predict the suitabiliity of 800K different molecules for use in solar panels. The relevant physical property of these molecules that we tried to predict was the "gap" value, defined by the difference in energy between the highest and the lowest occupied molecular orbital. To do this, we had access to a 100K samples training set. The features that we used were all computed from the string representation of the molecule, using a package called RDkit. We will first present how we preprocessed the data, and we will then describe the different machine algorithms that we used and the results we obtained.

1 Technical Approach: Preprocessing

1.1 Removing weakly correlated features

The samples that we were given contained 256 features precalculated using RDkit. The first step was to check how many of these features contain significant information, and to remove weakly correlated features. In theory, extra useless features should not prevent us from finding a good model, as they can easily be ignored if necessary (by setting their coefficient to 0 for a linear regression for example). However, they can slow down the tuning process of the model, and keep us from exploring some models or some non linear feature transformations within a reasonable time.
To filter the features that we judged useless, we first looked at their correlation with the output that we were trying to predict (the gap values), and got rid of every feature that was not at least 10% correlated. It appeared that the vast majority of the features were poorly correlated with our outcome of interest. We are well aware that in theory a feature could have an impressive predictive power while showing a 0% correlation with the output (for example if the correlation is +1 on half of the samples, and -1 with the other half, which a decision tree could exploit if these two halves can be separated using the value of one of the predictors), but we find it unlikely here and we are willing to take the risk to loose some information. After this first preprocessing step, we were left with only 7 predictors. Note: to gain some time, we only used 20% of the data to determine which features we should use.

1.2 Adding chemistry features with rdchem

From basic quantum mechanics, there are tree kinds of information that will determine the gap: first, the degree of electron orbital overlap (bonds), second the size of the nuclei, and third the chemical potential level, which is related to the average number of electrons per atom. We found the following usefull properties the rdkit Descriptor package: number of heavy atoms, number of atoms, molecular weight, average molecular weight of heavy atoms, average molecular weight, number of valence electrons and number of radical electrons.

RMSE comparison with a benchmark:

- Without extra features: LR RMSE = 0.3047; RF RMSE = 0.2874
- With extra features:LR RMSE = 0.2252; RF RMSE = 0.2273

1.3 Adding nonlinear terms

To capture the potential interactions between the different predictors, we added the interaction terms (the products between the different predictors). We also computed the quadratic terms. This was only a realistic idea because of the previous filtering of the not so useful features.

1.4 Scaling

Not all the features have the same order of magnitude. Because it is not intuitive that the unit in which a feature is expressed should influence its ability to affect prediction, and because we want all the features to be equally able to affect the prediction, we normalized all our predictors. (Features with a large range affect some models more than features with smaller range. Decision trees are not affected, but linear regression is, for example.)
We made sure to do that by only looking at the training set, and then applied the same transformations to the columns of the testing set.

2 Technical Approach: Models

2.1 Main model: Regularized Linear Regression

We chose to focus on Linear Regression, and regularized it in different ways.
We first used LASSO as a penalty: LASSO adds a regulariation term in norm 1, which encourages a sparse model: the total sum of the coefficients tends to remain small, but it does not matter how this quantity is spread between the features. So if some features are correlated, one of them will tend to carry the weight of the others and see its coefficient increase, while the other features see their coefficient set to zero.
We then used RIDGE regression: this time the penalty term is in norm 2, so big coefficients are even more penalized: not only is the sum of the coefficient encouraged to remain small, but individual coefficient tend to do the same. So if several features are correlated, the model will tend to distribute the total amount of coefficient between the features. Therefore the resulting model is not as sparse as the one generated by LASSO regression.
Finally we used an elastic net. An elastic net is basically the combination between a LASSO and a

RIDGE regression, since an extra hyperparameter is introduced: if this parameter is 0, the elastic net is equivalent to a LASSO. If this parameter is 1, it is equivalent to a RIDGE. For every value in between 0 and 1, the model is trying a weighted compromise between LASSO and RIDGE regularization. So an elastic net basically introduces an extra hyperparameter to do model selection between LASSO, RIDGE, and their combinations.

2.2 Partial Least Squares

Since we are dealing with a large sparse matrix (many zeros in the original features), we decided to explore dimension reduction. In particular, we focused on the Partial Least Squares (PLS) algorithm, which projects the data onto planes of greatest variance. These planes are referred to as principal components, and they represent the degree of variability that exists in the data. We have already seen in the Preprocessing section that not all features are equally important in predicting the data.

Our results are plotted in figure 1. The three plots correspond to PLS results different sets of data: (a) original data, (b) original data with additional chemistry features, and (c) truncated data with chemistry features, and (d) truncated data with additonal features and polynomial terms. For comparison, all three tests were performed on the same subsets of the data, with 40,000 molecules used for training and 8,000 molecules used for testing.

All four plots show the RMSE converging toward a minimum value, but the value of the RMSE plateau is different in different cases. With only original data, the error is 0.3, comparable with the LR benchmark. This indicates to us that the features in the data can be accureately represented by a little over 10 components. However, the plateau in RMSE does not mean that we have found the optimal model. It merely indicates that there may be non-linear features present in the data, which we cannot identify with a principal component analysis.

Truncated data set with additional chemistry features produces improved results, shown in Fig. 1 (c), and the RMSE converges after 30 components. This confirms to us that chemistry features do in fact provide new information not present in the original data. In order to introduce nonlinear terms to the data, we took squares, inverses and inverse squares of non-binary features. The reasoning is that the energy gap could be inversely proportional to one of the quantities. For example, the Bohr atom energy is inversely proportional to the energy level number, although there is no direct analog to that in our features. The results of PLS applied to the data including the new powers of numeric features are shown in Fig. 1 (d). This approach yielded the lowest RMSE among all PLS methods. We also note that it takes 40 principal components for the RMSE to converge. This number is greater than that in (a) - (c), which means that the new components we added are in fact significant. We concluded that reducing the number of features was justified, and that there is some non-linearity present in the data.

2.3 Multilayer Percetion (MLP) Regression

We made a few attempts at using the MLPregressor from the sklearn neural network module to fit out data. We applied MLP to the original data as well as the truncated, feature-added data. In both cases, we saw an increase in RMSE with increasing number of hidden layers, with the minimum around 2-3 layers. In diagnosing the issue, we saw that the cross-validation loss would decrease rapidly at first and then stop changing, possibly trapped in a local minimum. Suspecting that we may not be finding the global minimum due to numerical issues, we have tried several

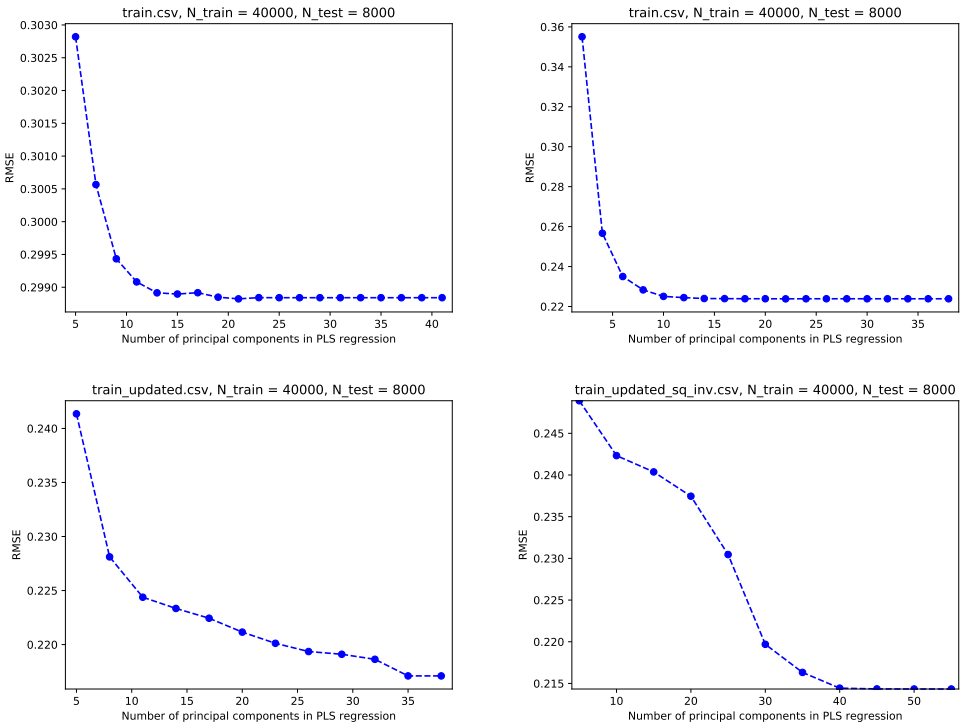

Figure 1: (a) Results of PLS trained on the original data (no extra features). The RMSE is plotted as function of the number of principal components specified in the fitting. (b) Similar to (a), but but after adding rdkit features to the original data. (c) Similar to (a), but using the truncated dataset with chemistry features. (d) PLS on the truncated data set with additional features, their squares, inverses, and inverse squares. This is the best RMSE among our PLS attempts

steepest gradient descent-like methods, including a stochastic method with adaptive and constant step sizes. We then tried several activation functions, which penalize outliers differently. Finally, we tried to force the method to keep iterating by increasing the maximum number of iterations and decreasing the tolerance, with little success.

3 Methods: Validation (should we remove this?)

3.1 Split

In order to develop and validate our models quickly without having to submit them on kaggle, we used smaller subsets of our data (typically below 100,000 molecules) for testing models. We would use 20 percent of this subset as testing data, and the rest as training data. We would then study the error as function of the method used.

4 Discussion

Given our results so far, the most promising direction to pursue is feature engineering: excluding insignificant features and adding new features made a significant improvement in our accuracy. In addition, methods that worked well involved some use of randomness or nonlinearity or both. These conclusions agree with an intuitive interpretation of the results in Fig. **??**. On the other hand, multilayer perception did not give immediate results. Given the time constraints, it is possible that we have not identified the correct neural net-like algorithm.

Given more time, we would pursue new features combined with better tuning of our models, most notably Random Forrest algorithms.