Practical 1: Predicting the Efficiency of Organic Photovoltaics

Writeup due 23:59 on Friday 13 February 2015
Kaggle submission closes at 11:59am on Friday 13 February 2015

You will do this assignment in groups of three. You can seek partners via Piazza. Course staff can also help you find partners. Submit one PDF writeup per team via the Canvas site.

Competing on Kaggle: You are expected to submit at least one set of predictions to the Kaggle competition online at

https://inclass.kaggle.com/c/cs181-practical-1-predicting-the-efficiency-of-photovoltaic-molecules

You should make your Kaggle submissions as a team. You should be a part of exactly one team. Do not make submissions separately from your team, and please use your real name for your user identity so that we can identify your results. There is a limit of four submissions per day, where "day" is determined by UTC. (Your deadlines are still in eastern time.) **Note that the Kaggle submission site closes 12 hours before the Canvas dropbox. This is to ensure that you are able to write up any last-minute submissions.** You should be able to join the competition by registering with your university email address. If you have trouble joining the contest, please email the staff list.

The Harvard Clean Energy Project

What if you could capture and convert sunlight into electricity with a material as cheap and as versatile as a plastic bag? What if the material could be produced on a massive scale, with easily accessible technology? What if other versions of the material could be coated, painted, or sprayed on building surfaces for solar energy production? What if these materials were ultra-thin and ultra-light for portable devices? And finally, what if they were inexpensive and could provide electricity to people in the developing world? – Harvard Clean Energy Project

Solar power is one of the most promising technologies for renewable energy to reduce our dependence on fossil fuels. Unfortunately, most modern solar cells are based on silicon. These materials are rigid, expensive, and difficult to manufacture. On the other hand, *carbon* based solar cells could be cheap to produce, flexible, transparent, and be made and molded as easily as plastics. There's just one catch: no known organic photovoltaic molecules are as efficient as their silicon counterparts.

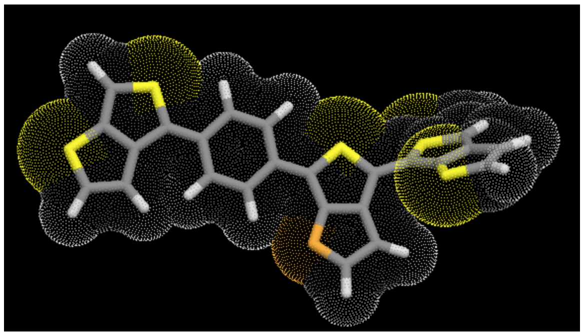
The Harvard Clean Energy Project has been using massive scales of computation to explore new possibilities for organic photovoltaics. The project uses density functional theory (DFT) to estimate the the properties of the molecules that determine their potential efficiency as solar cells. The main quantity of interest is the difference in energy between the highest occupied molecular orbital (HOMO) and the lowest unoccupied molecular orbital (LUMO). It can take hours or days to compute this accurately on a modern computer.

Recently, it has become clear that machine learning might have something to say about this. It may be possible to sidestep these expensive DFT computations by learning a function from a feature representation of the molecule to the HOMO-LUMO gap. From our point of view in CS181, this is a regression problem: take molecular features and produce a real-valued prediction of what the DFT would calculate. Better machine learning models for this problem could lead to new kind of materials and more efficient solar cells!

You have one million molecules to train on, and are tasked with making predictions on another 800,000 or so. You'll upload your predictions to Kaggle, where a subset will be used to produce a "public leaderboard" and another subset will be used to reveal the final rankings at the end of the contest. You'll have access to complete information about the molecular structure in the form of a SMILES string. This is a representation that chemists like to use to encode molecular structures. Here is an example:

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

This SMILES string is a molecule that looks like this:



One of the challenges of making predictions about molecules is finding a good feature representation. Clearly, the SMILES string itself isn't going to be all that helpful. Luckily, many chemists have built tools to build interesting representations. One such tool is a Python package called RDKit. With it you can easily load SMILES strings and produce features. Alternatively, you can just use the features we've already extracted with RDKit, which produced 256-dimensional binary vectors. These vectors are in the training and test files provided.

Data Files

There are three files of interest, which can be downloaded from the course website and also from Kaggle:

- `train.csv.gz` – This file contains information about one million molecules. It has 258 columns. The first column is a SMILES string, which is a representation of the molecule in case you would like to see what it looks like or build better feature representations. The next 256 columns are a binary feature representation that RDKit outputs. These are features of the molecule that you might use to build a regression model. The final column is the difference in energy between the HOMO and the LUMO. This is the label you are trying to predict.

- `test.csv.gz` – This file contains another 800,000 or so molecules. In this case, the label is not provided. These are the ones you are trying to make predictions for. The first column is a numeric identifier that you should use when constructing your prediction file to upload to Kaggle.
- `example_mean.csv` – This is an example of a prediction file, such as you might upload to Kaggle. The id numbers match those in the test file, but the predictions are very naïve: just the mean of the training data.

Evaluation

After you upload your predictions to Kaggle (which you can do at most four times per day), they will be compared to the held-out true HOMO-LUMO gaps, as determined by density functional theory calculations. The score is computed via root means squared error (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

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

Sample Code

A Python file is available from the course website to help you get going. The file `example_mean.py` is a simple script to load in data and produce the `example_mean.csv` prediction file. You don't need to use this file; feel free to build the prediction system as you see fit.

Solution Ideas

You have a lot of flexibility in what you might do. You could focus on feature engineering, i.e., coming up with fancy inputs for your method using a tool like RDKit (or your own knowledge of chemistry!), or you could focus on fancy regression techniques that use the features we provide. Here are some ideas to get you started:

- **Ridge Regression:** You could use a simple L_2 regularization approach to regression weights and use cross-validation to determine an appropriate regularization penalty.

- **Lasso Regression:** You could got a bit fancier and use L_1 regression to identify a sparse solution, again using cross-validation to determine an appropriate regularization penalty.
- **Elastic Net:** Use both L_1 and L_2 at once!
- **Neural Network:** Get a jump on the course material and build a neural network to make predictions. Explore the world of deep learning!
- **Support Vector Regression:** Prefer your problem convex? Get a jump on kernel methods and build a support vector machine for regression.
- **New Feature Ideas:** Try out some of the different fancy feature representations and chemical fingerprints that different cheminformatics tools provide. Maybe there's even a better way to use RDKit to extract features.
- **Go Totally Bayesian:** Worried that you're not accounting for uncertainty? You could take a fully Bayesian approach to linear regression and marginalize out your uncertainty.
- **Go Nonparametric Bayesian:** Intrigued by the infinite-dimensional machine learning? Check out Gaussian process regression.

Questions and Answers

What should I turn in via Canvas? The main deliverable of this practical is a three-to-four page typewritten document in PDF format that describes the work you did. This may include figures, tables, math, references, or whatever else is necessary for you to communicate to us how you worked through the problem and ultimately how you made your best predictions. Make sure to include the name of the team (as it is on Kaggle) and the name of all partners.

How will my work be assessed? This practical is intended to be a realistic representation of what it is like to tackle a problem in the real world with machine learning. As such, there is no single correct answer and you will be expected to think critically about how to solve it, execute and iterate your approach, and describe your solution. The upshot of this open-endedness is that you will have a lot of flexibility in how you tackle the problem. You can focus on methods that we discuss in class, or you can use this as an opportunity to learn about approaches for which we do not have time or scope. You are welcome to use whatever tools and implementations help you get the job done. Note, however, that you will be expected to *understand* everything you do, even if you do not implement the low-level code yourself. It is your responsibility to make it clear in your writeup that you did not simply download and run code that you found somewhere online.

You will be assessed on a scale of 20 points, divided evenly into four categories:

1. **Effort:** Did you thoughtfully tackle the problem? Did you iterate through methods and ideas to find a solution? Did you explore several methods, perhaps going beyond those we discussed in class? Did you think hard about your approach, or just try random things?
2. **Technical Approach:** Did you make tuning and configuration decisions using quantitative assessment? Did you compare your approach to reasonable baselines? Did you dive deeply into the methods or just try off-the-shelf tools with default settings?
3. **Explanation:** Do you explain not just what you did, but your thought process for your approach? Do you present evidence for your conclusions in the form of figures and tables? Do you provide references to resources your used? Do you clearly explain and label the figures in your report?
4. **Execution:** Did you create and submit a set of predictions? Did your methods give reasonable performance? Don't worry, you will not be graded in proportion to your ranking; we'll be using the ranking to help calibrate how difficult the task was and to award bonus points to those who go above and beyond.

Bonus Points: The top three teams will be eligible for extra credit. The first place team will receive an extra five points on the practical, conditioned on them giving a five-minute presentation to the class at the next lecture, in which they describe their approach. The second and third place teams will each receive three extra points, conditioned on them posting an explanation of their approach on Piazza.

What language should I code in? You can code in whatever language you find most productive. We will provide some limited sample code in Python and can also provide some support for Matlab. You should not view the provided Python code as a required framework, but as hopefully-helpful examples.

Can I use {scikit-learn | pylearn | torch | shogun | other ML library}? You can use these tools, but not blindly. You are expected to show a deep understanding of the methods we study in the course, and your writeup will be where you demonstrate this.

These practicals do not have conceptual questions. How will I get practice for the midterms? This is the role of the homeworks. We will also provide practice problems and solutions in section. You should work through these to help learn the material and prepare for the exams. They will not be a part of your grade.

Can I have an extension? There are no extensions to the Kaggle submission and your successful submission of predictions forms part of your grade. Your writeup can be turned in up to a week late for a 50% penalty. There are no exceptions, so plan ahead.

Find your team early so that there are no misunderstandings in case someone drops the class.

Changelog

This format for assignments is somewhat experimental and so we may need to tweak things slightly over time. In order to be transparent about this, a changelog is provided below.

- **v1.0** – 4 February 2015 at 1:00pm
- **v1.1** – 8 February 2015 at 1:15pm: Fixed two typos that said MAE which should've been RMSE.