Machine Learning for Molecular Engineering
3/7/10/20.C01 (U) 3/7/10/20.C51 (G)
Spring 2024

Problem Set #6

Date: April 24, 2024
Due: May 10, 2024

Instructions This is the final problem set for the undergraduate version of this course (3.C01, 10.C01, 20.C01). This exercise includes a machine learning competition hosted on Kaggle and a short preliminary exercise about molecular representation and missing data imputation. Here is the link to the competition. For submission, you will need to submit a code notebook containing the code and a short writeup to describe your solutions. You can find the template notebook here. You can also submit a separate notebook as the solution to the machine learning competition. **Because this is the final pset, you will need to use your discretion in deciding how to tackle these problems; we have not specified exactly how each part should be approached.**

Background

The development of novel and effective oncology drugs requires extensive testing of cancer cell lines to better understand how drugs work. However, traditional phenotypic screening methods only test one cell line at a time, which is time-consuming and resource-intensive. Moreover, data generated from a small subset of cell lines may result in an incomplete understanding of the drug's efficacy. To address these challenges, novel multiplexed cancer cell line screening methods have been developed.

In particular, for this problem, we will consider PRISM - a technique pioneered at the Broad Institute which employs a unique DNA barcode-based approach to accelerate the screening process. [1] All perturbations screened in our datasets will be small molecule agents already approved by the FDA for human consumption and therefore ripe for repurposing.

Registering for the competition

Go to kaggle to register for an account. After you register successfully, click here to join the competition. You can submit your solutions to the test set and your result will be displayed on the leaderboard. If you prefer anonymity, you can choose a nickname like `molmlmaster` rather than your real name; if you do this, please include your nickname in your final writeup. Note that you are not graded based on your ranking but a combination of different factors which will be described below. Our competition will have no cash prizes, but you can explore other competitions on Kaggle for potential cash prizes and employment opportunities.

Part 1: Predicting Cell Line Viability from Perturbations

In this problem, we will model the survival of 568 human cell lines under 4000+ small molecule perturbations. The cell lines are selected across all tissue types and represent primary, cancer, and immortalized lineages. Training a predictive model of the impact of small molecules on cell state is an important question in drug development, as one might imagine, where in many diseases (e.g.,

cancer) the goal is often to optimize a selective chemical treatment to target specific diseased cell phenotypes.

Part 1.1 (3 points) Morgan Fingerprint Representation

In order to predict the effect of unseen chemical perturbations, first we need to represent the chemical features of our perturbations. For this we will use Hashed Morgan Fingerprints as implemented in RDKit [2]. The SMILES representation of each perturbation molecule is given in `prism_train.csv`. Compute the Morgan Fingerprints for each of the chemical perturbations and store them as your preliminary X representation. Later, for your own implementation you're tasked to consider other ways of molecular representation.

Part 1.2 (2 points) Separate Dataset into Splits

Split the `prism_train.csv` into test and validation splits (80:20). As is familiar by now, this is necessary for model training and hyperparameter tuning.

Part 1.3 (4 points) Missing Data Imputation

In the biological problem setting often datasets will include missing values where either the experimental conditions failed, instrumentation lacked requisite sensitivity, or where data collection was simply out of scope. For these cases it is common to employ data imputation strategies (e.g., matrix completion algorithms) to fill in missing data values as a preliminary step before fitting downstream models to the data. In our case, the PRISM dataset does indeed contain missing cell values and we will be filling them with nearest neighbor (KNN) imputation. Conceptually simple, KNN imputation compares rows of a dataset, filling missing values using the k rows which are most similar based off the values which are not missing. Still it is seen that KNN imputation generally performs on par with more sophisticated techniques. [3] You will perform KNN imputation using cell line similarity as the comparative metric - as cell lineage is often the greatest factor in prediction of cellular responses.

Part 2: Baseline Prediction of Cell Viability

Using the Morgan Fingerprint representations and the imputed dataset from the previous part you will train two baseline models to get started. You don't need to submit your predictions to Kaggle for this part.

Part 2.1 (8 points) Train a Nearest Neighbor Regressor

Train a baseline nearest neighbor regressor to predict cell viability across cell lines for new chemical perturbations based on the cell viability scores provided (`prism_train.csv`). Show your code and report the root-mean-squared-error results of a 5-fold cross validation (mean and standard deviation). Note: be careful when benchmarking against your held out test set from `prism_train.csv` which should include missing values - these missing values should just be skipped when calculating RMSE.

Part 2.2 (8 points) Train a Neural Network Regressor

Train a 568 output neural network regressor to cell viability based on the same features. Show your code and report the RMSE results of a 5-fold cross validation (mean and standard deviation).

Part 3: (50 points) Machine Learning Competition and Report

In this part, you will apply techniques you have learned in this class to propose machine learning solutions to a classification problem. Based on the training data we provided to you, you will try to make predictions for the held-out test data. We have provided the test perturbation SMILES in `prism_test.csv`. You will use your model to predict log fold viability score across all 568 cell lines using this test feature set. The evaluation of test performance will be performed by Kaggle, and you can see your results in the leaderboard. The goal is find the best model possible.

You can submit up to 20 solutions per day. We have provided a utility function for you to generate a submission file. For evaluation, the metric we use is root-mean-squared-error (where missing test values are skipped), ranging from 0 to 1. The reported public score on Kaggle is calculated with only 50% percent of the test data, but your performance will be graded on the private score based on the other 50% of the test data; this is to prevent you from tuning performance to the test data (which is bad practice!). We will release the final performance values when the competition is over.

You need to submit your commented code (as a `.ipynb` notebook) with a writeup that addresses the following points:

1. Data preprocessing and representation

   For full credit, you are expected to try at least one additional representation method for each perturbation other than the simple hashed Morgan fingerprint representation outlined in Part 1.1. Try to apply what you have learned in the class as well as your own personal background expertise. Likely perturbation metadata (e.g., dosage and mechanism of function) and cell line representations will prove enriching. This data can be found in the supplementary dataframes `prism_perturbation_metadata.csv` and `prism_cell_line_metadata.csv` respectively. If you decide to adopt a representation or resource from an outside source, please include the reference in your writeup.

2. Choice of model architecture

   You are expected to try at least two models (in addition to the two baseline models in Part 3) and select the best-performing model to submit the best solution. If you are designing a novel model in PyTorch, describe your model architecture. For all your models, report cross-validation scores or some other rigorous metric of performance.[1] You need to provide a brief description of your model choice or design and mention any open-source code/packages you used. If you decide to adopt a model architecture or method from a paper, please include the reference in your writeup.

3. Model evaluation and selection

   Describe how you performed hyperparameter search. Describe how the model performance is evaluated to select the best hyperparameter.

[1] If you have a very good model but cross-validation is too compute-intensive, you can report a test set performance instead.

Grading Rubric

Here we describe how we will grade your notebook submission.

(25 points) Creativity

There are many modeling choices for multi-regression questions like this. You can use nearest neighbor and neural network regressors from part 2. You can also use support vector machines, gradient boosted trees, and matrix completion algorithms (e.g., joint row & column encoders). You are encouraged to survey the literature for inspiration. In [4], you can find some possible model types to apply to this problem. **Warning:** You should not merely copy publicly available code.

We encourage you to apply PyTorch in your solutions. With PyTorch, you can build more complex models that is optimizable by gradient descent. For example, you can write an MLP architecture augmented with attention mechanisms which has been very popular these days.

Likewise, we encourage to build rich representations for the molecular perturbations and for the cell lines which incorporate tissue, genetic, or transcriptional features. A good place to start would be the perturbation and cell line metadata files included in the kaggle challenge - there you can find crucial information such as perturbation dosage and cell line tissue lineages. But you're welcome to draw from any other publicly available resources. The sky is the limit!

(35 points) Technical correctness

We will examine your code and text to evaluate your solution on technical correctness and the appropriateness of your chosen methods. Please comment and document your code so that we understand your approach to data preprocessing, train/validation/test split, hyperparameter optimization, and cross-validation. It is always important to compare your model performance to that of appropriate baseline methods. While we don't expect every idea you come up with to outperform the baseline, we do expect you to make this comparison and accurately report the resulting performance of your models.

(15 points) Model Performance (10 + 5 points)

We have provided a nearest neighbor regression baseline and 3 additional TA-prepared models as benchmarks. The nearest neighbor baseline is similar to something you might have done for Part 2.1. You can earn up to 10 points plus 5 bonus points on this part, depending on your performance as follows:

1. Beat or tie the KNN baseline - between 2 and 6 points.
2. Beat or tie 1 TA benchmark - between 6 and 10 points.
3. Beat or tie 2 TA benchmarks - 10 points, with up to 5 bonus points possible.
4. Beat or tie 3 TA benchmarks - 10 points plus 5 bonus points; potentially more bonus points if you impress us.

Warning: This dataset is rather noisy, so your ranking may change considerably between the public and private scores. Be very careful to make sure you don't overfit.

References

[1] Corsello, S. M. *et al.* Discovering the anticancer potential of non-oncology drugs by systematic viability profiling. *Nature Cancer* **1**, 235–248 (2020).

[2] Morgan, H. The generation of a unique machine description for chemical structures—a technique developed at chemical abstracts service. *J Chem Doc* **5**, 107–113 (1965).

[3] Jager, S., Allhorn, A. & Biermann, F. A benchmark for data imputation methods. *Fontiers of Big Data* **4** (2021).

[4] Kourou, K., Exarchos, T. P., Exarchos, K. P., Karamouzis, M. V. & Fotiadis, D. I. Machine learning applications in cancer prognosis and prediction. *Computational and structural biotechnology journal* **13**, 8–17 (2015).