Final project, CSCI 471/571 Fall 2020

Description of tasks, datasets, and guidelines
version 2

Due: December 18, 11:59 p.m.

1 Overview

Read this whole document.

This project tests your machine learning skills on a number of datasets. Your job is to select a model that will generalize well to the testing sets in these supervised problems. Linear models will not perform well but can be a useful benchmark for comparison. You should be using more advanced techniques that we have talked about but not extensively studied in the homework. You are required to submit predictions for the labels in the test set along with predictions for the generalization error you expect to incur on the test set.

A small part of your grade will be based on the ranking of your predictions against your classmates.

You are allowed to employ *any libraries you wish* to obtain good accuracy. The only requirements are that the libraries be freely available for download and you follow the terms of use. Recommendations include:

- `scikit-learn`, for general-purpose machine learning
- `tensorflow`, `pytorch`, `keras`, for neural networks
- `xgboost`, gradient boosting
- `falkon`, for large-scale kernel learning (`https://falkonml.github.io/falkon/`)
- `neural-tangents`, neural tangent kernels (`https://github.com/google/neural-tangents`)
- `weka`, another general-purpose library built in Java

I want you to **work alone** to produce your own pipeline and write your report. However, you may help each other to use whatever libraries you end up using. Therefore, it is okay for two students to discuss how to use the routines in a library, share documentation, or link to helpful websites. However, it is not okay for two students to share code and build very similar pipelines.

2 Supervised learning tasks and datasets

Data is provided on the Google Drive.

2.1 Dataset 1

This is a regression problem. The training data consists of 27,930 samples, and you are required to provide predictions on 9,311 test samples, for which you do not have the labels. The features are 9,491 dimensional and integer-valued. The labels are real scalar numbers.

Each sample is a different chemical, and the features describe important physical properties of those chemicals. The labels are binding affinity to a molecule of interest.

The evaluation metric is the median absolute error `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.median_absolute_error.html`.

2.2 Dataset 2

This is a classification problem. The training data contain of 463,715 samples, and the test data contain 51,630 samples. The features are 90 dimensional and real-valued. The possible labels are 0 (36,487; 8%), 1 (149,746; 32%), 2 (277,482; 60%), where the values in parentheses denote the number of samples with each label and their proportions.

Each data point is a song, and the features are related to how much power there is in different frequencies throughout the song. The labels are something to do with the year the song was released.

The evaluation metric here is the balanced accuracy score `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html`. This takes into account the fact that the classes occur in different proportions, making them "imbalanced." In the test set, the labels will occur with similar proportions.

3 Suggestions

It is a good idea to experiment with many different models to see what works well for a given problem. You should also consider different preprocessing techniques such as standardization, scaling, feature selection and transformation.

A big part of making sure your model generalizes well is being careful about the model selection step, where you evaluate models and tune hyperparameters. Be sure to use cross-validation techniques to select hyperparameters and estimate testing error on unseen data. For hyperparameter search, I recommend performing what's called a grid search, where you pick a range of possible parameter values to fit to training data. Since you will have to fit many models, this step can be computationally quite costly, so it's in your interest to start running your models sooner than later. Compare your models using the evaluation metrics for those problem rather than the model's intrinsic loss function.

The classification problem is a multiclass problem with imbalanced classes. The `scikit-learn` classifiers all have multiclass support built-in. Here you can find some suggestions for working with imbalanced data: `https://www.kdnuggets.com/2017/06/7-techniques-handle-imbalanced-data.html`. Consider using an under-sampling or ensembling method during training so that your model learns how to classify the infrequent examples. There is a contrib package called `imbalanced-learn` that looks very useful and has good documentation: `https://imbalanced-learn.org/`.

4 Grading

The majority of your grade (≈90%) will be based off the report you write up explaining what you did. The rest of the grade (≈10%) will be based on the performance of your predictions relative to your classmates. You are required to submit both predicted labels for the test set and an estimate of the test error you will incur on those labels. Your score will reflect both kinds of predictions. The undergrad and grad sections will be evaluated separately. If everyone's models have high accuracy, it won't count against you.

5 Project report

Each student will write up a project report describing how you analyzed the data, selected and fit a model, and produced your predictions. The report should include an introduction, description of the methods used, results from applying those methods including plots or tables that describe those results, and concluding remarks. You are not required to stick to a rigid format, but an example could be:

1. Introduction
2. Dataset 1
   (a) Methods
   (b) Results

3. Dataset 2
    (a) Methods
    (b) Results
4. Conclusions

Since you will probably be employing methods that we have only touched upon in this course, you should explain how each method works. A good level of detail would include the mathematical formulation, the algorithmic implementation (e.g. "the cost is minimized with gradient descent"), and an explanation of what kind of problems for which the method is most suitable. You can assume that the reader has a general level of machine learning understanding.

Your report will evaluated on:

- Soundness of approach and application of methods
    - Do you present good justification for the models and training algorithms you used?
    - Did you explain, at an appropriate level of detail, the gist of how the algorithm works?
    - Were you using the models appropriately?
    - Did you use regularization appropriately?
    - If your technique benefits from feature pre-processing, did you do that?
- Quality of analysis
    - Did you tune the relevant hyperparameters?
    - Did you provide a clear illustration of the hyperparameter tuning (with figures or tables)?
    - Did you discuss the observed trends and offer plausible explanations for them?
    - Did you use the tuning results to select your model for system evaluation?
- Quality of exposition
    - Is the report cohesive?
    - Is the report free from typos and grammatical errors?
    - Is notation consistent? Are all (non-standard) symbols or acronyms defined?
    - Are the figures clear and easy to interpret (appropriated labeled, etc.)?
    - Is the general appearance professional (properly typeset mathematics, etc.)?

6 Other guidelines

You may:

- Increase the dimensionality of the features (e.g. polynomial or random features)
- Decrease the dimensionality of the features (e.g. PCA)
- Use your features as is
- Discard some of the training data points
- Manufacture new data points from the provided data points (e.g. by adding random noise, doing transformations, etc.)
- Train your own models using any free machine learning toolkit (subject to licensing terms)
- Use any code you have written for this or previous classes to train your models

- Heavily tune your models to find the optimal hyperparameters
- Use machine learning algorithms even if they were not covered in this class
- Invent your own machine learning algorithms and use them
- Share tips and model suggestions with other students
- Run `nice` processes on any of the computers in the department's general purpose computing labs (CF 162, 164, 165, 167, 405, 418, or 420)
- Use your personal computer(s) for this task

You may not:

- Obtain and use any new training data not directly derived from the training data files provided to you (e.g. by search the web, buying corpora, etc.)
- Share code with any other student
- Share any test set predictions with any other student
- Share any part of your report writeup with any other student
- Distribute the models, data, or predictions to anyone not enrolled in our class
- Obtain models, data, or predictions from anyone not enrolled in our class
- Use computing resources not listed in the previous section. This prohibits, among other things, using the department compute cluster, cloud computing services, and computers in department research labs (e.g. CF 408, 498, etc.)
- Use an excessive amount of department computational resources, to the extent it interferes with the normal operation of lab machines
  - To avoid excessive memory usage, do not run processes that utilize more than 80% of the physical memory on machine
  - To avoid excessive CPU usage, you must run all of your jobs with a nice of 15
  - To avoid excessive file and network I/O, avoid redundant reads or writes, try to write to local disk (/tmp) whenever possible, and avoid running too many processes simultaneously. For long-running processes, make sure you're closing file handles, freeing memory and explicitly triggering garbage collection if needed.