ece5307 Sp23 Final Project

Phil Schniter

Due April 25, 2023 @ midnight

1 Introduction

This project tests your understanding of how to design predictors for regression. The data comes from a real-world application, but the application details are not important.

The training data has been made available to you in a starter GitHub repository, similar to how the labs are distributed. There, `Xtr.csv` contains training features and `ytr.csv` contains training targets. Some important characteristics of the training data are:

- number of targets: $K = 1$
- number of training samples: $n = 10000$
- feature dimension: $d = 26$

As the performance metric, we will use the R^2 coefficient of determination. The R^2 value of your designs will be evaluated using Kaggle, as described below. During the design stage, you can evaluate the R^2 of your models using the `r2_score` function in `sklearn`.

2 Design approaches

The main goal of the project is to *demonstrate your understanding of the principles/methods taught in ece5307.* If you implement super-powerful methods from outside the course, but don't demonstrate your understanding of what was taught in our course, you won't score a high grade.

Each 3-person team should attempt each of the three main approaches to regression that we discussed in this course: 1) linear (including linear regression, linear SVR, kernel SVR, and their regularized counterparts), 2) tree-based (including Random Forest and XGBoost), and 3) neural-net. It's suggested that each team member team tackles one of these three approaches, but you can work together if you like. Each 2-person team can choose to tackle any two of these three approaches. In any case, *your written report must indicate which team members worked on which design tasks.*

The following are a few design ideas to consider for each approach. You are not limited to the ideas below, nor do you have to tackle every idea below. Points will be awarded based on how much work is done and how well that work is done.

2.1 Linear

- Apply linear regression to this regression task.
  - What kind of standardization works best?

  - Does regularization (e.g., L2 or L1) help? What is the best regularization weight?
  - Does feature selection help?

- Apply support vector regression (SVR) to this regression task. (Note: the SVR is very similar to the SVC, but it applies to regression rather than regression. It shouldn't be a problem that we didn't discuss it in detail.)
  - Similar questions as logistic regression, plus...
  - What is the best kernel? What are the best kernel parameters?

2.2 Neural networks

- Apply a multilayer perceptron to this regression task.
  - What is a good choice of # layers?
  - What is a good choice of # hidden nodes in each layer?
  - What is a good choice of hidden activation functions?
  - What are good choices of learning rate and/or learning-rate schedule?
  - Do batch-norm and/or dropout help?

- Apply a convolutional deep network to this regression task.
  - Similar questions as the multilayer perceptron, plus...
  - What is a good choice of # channels in each layer?
  - What is a good choice of kernel size?

2.3 Tree-based methods

- Apply random forests to this regression task.
  - What are good parameter choices?
  - Does feature selection help?

- Apply XGBoost to this regression task.
  - Similar questions as random forests.

3 Kaggle

Kaggle will be used to score your attempts at each of the above three approaches. For this purpose, 3 private competitions have been created:

1. linear: `https://www.kaggle.com/t/7c8da2e65f16409282bb086812837fb5`
2. tree-based: `https://www.kaggle.com/t/ee8ba9ab57384791a17b3809aa0d2a4b`
3. neural-net: `https://www.kaggle.com/t/997b042765734609baa29c09ffd9d9b8`

To use kaggle, you will need to create a (free) account if you do not already have one. For these competitions, you will make predictions of the targets associated with the test features in `Xts.csv`, which is included in the starter repo. You will then export those predictions in a properly formatted .csv file and upload them to kaggle for scoring. To generate those .csv files, you can use the `validation.py` script provided in the project repo (see below). Kaggle will show your test R^2, as well as the test R^2 attained by other teams. Important:

- Your kaggle rank will affect the grade of your final project. For each team, we will consider only the single best design in each of the three contests.
- Enter only linear results into the linear competition, neural-net results into the neural-net competition, and tree-based results into the tree-based competition. Violations will result in a loss of points!
- Enter your team number (as found on Carmen → People) in the kaggle Team page, since your kaggle user name may be unrecognizable to the grader. Please don't submit entries with your personal name.
- You are allowed at most two kaggle submissions per day, so use them widely, and don't wait until the last minute.
- The reported performance is actually the performance on the "public" fold of the test data. There is also a "private" fold, which only the instructor and TAs can see. The two folds were randomly generated, so they should behave similarly.
- Given the models you've submitted, kaggle is supposed to choose the one with the best public-fold performance for final scoring on the private fold. However, it doesn't always seem to do this, so I recommend choosing it manually via the My Submissions interface.
- The following website may be useful: `https://www.kaggle.com/c/about/community`

4 Submission of code, model, and standardized data

You will need to submit the following using GitHub (similar to the labs):

- Your Jupyter notebooks (used to design, train, and analyze each of your models),
- your trained models (saved in the proper format, as described below), and
- your standardized datasets.

The latter two components will be used to verify that your models generate the targets submitted to kaggle. Because the model-saving procedure differs between sk-learn, XGBoost, and PyTorch, we have provided a `training_demo.ipynb` for each case, which you can modify as needed. Those demo notebooks also show how to use the `validation.py` script to output a kaggle-formatted .csv file from your saved model and standardized data. *Remember to commit all of these files to GitHub so that the grader can access them!*

If GitHub complains that your saved model is too large to commit, try zipping it to see if that helps. If not, please upload it to a site like DropBox and inform the professor.

You are welcome to re-use code from the demos or labs from this course without citing it. *If you borrow code from another source, you must cite it.* Failure to do so will be considered plagiarism.

5 Project report

Each group must submit one written report in pdf form via Carmen. Team members can work together on the report, or they can write their own sections and merge them at the end. The goal of the project report is to describe *how you arrived at each of your final designs.*

- Explain the design process that led to your best method for each competition (i.e., linear, tree, and neural net). For example, explain how you chose your hyperparameters. Was it the result of cross-validation? If yes, please show the relevant results/plots. If you used feature selection, explain how you implemented it. And so on.
- The report should contain detailed results in the form of tables and/or figures. Little credit will be given to statements like "We tuned C using K-fold CV" when not accompanied by a clearly labeled plot/table of the performance versus C."
- Any time you discuss R^2 or "performance," make sure to specify whether it is training R^2, test R^2, cross-validation R^2, or kaggle R^2. And remember that training R^2 is *not* an appropriate metric for hyperparameter optimization.
- There's no need to include explanations of concepts that are already explained in the notes. For example, you don't need to explain logistic regression, SVR, neural networks, etc.
- Try not to include code in the written report. Rather, clearly explain what you did and why you did it using words, plots, and tables.

The written report should obey these formatting rules, else points will be deducted:

- maximum 15 pages for a 3-person group or 10 pages for a 2-person group, not including a list of references,
- double spaced,
- 11pt font,
- 1" margins on all sides.

The final report *must identify which portions of the project (writing and code) were tackled by which team members.* Each team member will get an individual grade for the project.

6 Grading Rubric

The final-project grade will be scored as follows:

25pts Technical approach: a) design of methods, b) evaluation of methods.

50pts Written report: a) description of methods, b) description of parameter tuning, c) description of results, d) analysis/interpretation of methods and results, e) references, f) organization, formatting, writing, grammar, spelling.

25pts Kaggle score: Your kaggle score will be curved to give a numerical grade between 0% and 100%. The top kaggle score will map to 100%, and the average kaggle score will map to 80%. All other kaggle scores will be linearly mapped according to this curve.