COL 764 Assignment 3 - Learning to Rank

2024 - Version 1.0

Version History

Revision	Date	Author(s)	Description
1.0	14 Oct 2024	Instructor	Initial Version.

Deadline

Deadline for final submission of the complete implementation, and the report on the algorithms as well as performance tuning: October 25th 2024, 09:00 AM.

Weightage

This assignment is evaluated against 60 marks. The tentative breakup of marks is given in the end of the document. There is an optional bonus part of 15 marks.

Instructions

1. This programming assignment is to be done by each student individually. Do not collaborate -either by sharing code, algorithm, and any other pertinent details- with each other.
2. All programs have to be written using Python 3.12.0, and scikit-learn 1.5.2 (latest stable release). Anything else requires explicit prior permission from the instructor. Please note that you are expected to install scikit-learn on your own. In case you face problems, feel free to post it on the Teams channel.
3. A single tar/zip of the source code has to be submitted. The zip/tar file should be structured such that
   - upon deflating all submission files should be under a directory with the student's registration number. E.g., if a student's registration number is `20XXCSXX999` then the zip submission should be named `20XXCSXX999.zip` and upon deflating **all contained files** should be under a directory named `./20XXCSXX999` only (names should be in uppercase). Your submission might be rejected and not be evaluated if you do not adhere to these specifications.
   - It is the responsibility of each student to ensure that the program executes correctly, and follows the name & argument structure precisely as defined in this document. If it does not execute on our test machine, then no marks will be awarded for that component. **Note that we will use only Ubuntu Linux machines to build and run your assignments.** So take care that your file names, paths, argument handling etc. are compatible.
4. You *should not* submit data files, index files, dictionaries etc. If you are planning to use any other "special" library, please talk to the instructor first (or post on Teams).
5. **Note that there will be no deadline extensions. Apart from the usual "please start early" advise, I must warn you that this assignment requires significant amount of experimentation, including tuning of parameters to get good speed up and performance. Do not wait till the end.**

1 Assignment Description

Data Releases

All datasets will be made available on HPC under the folder: `/scratch/cse/faculty/srikanta/COL764-A3-2024`. The sub-folder names under this folder will be self-explanatory.

In this assignment, you will be exploring the use of machine learning models to train, validate and evaluate learning to rank using a large benchmark collection of query-document features. Note that in this assignment, you are not required to do a lot of fresh "implementation", but you are required to do a lot of experimentation and result analysis. Therefore, your report carries a lot of weightage for evaluation.

1.1 Datasets

These are the two datasets for you to work with –

TD2003 & TD2004 Both of these are feature-ized versions of TREC Topic Distillation tasks from 2003 and 2004 respectively. The details about the dataset are in the attached supplementary document. There are 5-folds of data partitioned into `trainingset.txt`, `validationset.txt` and `testset.txt` each. We will release 4-folds for you to work with. We will run your submission files on the 5th fold to evaluate. You must train, validate and test your model on all the 4 folds, and report the performance in each fold.

1.2 Tasks

You are expected to make use of a mature machine learning library - scikit-learn `https://scikit-learn.org/stable/`. You will implement the **pointwise learning to rank** using

1. Support Vector Regressor (SVR)
2. GradientBoostedRegressor, and
3. MLPRegressor

The goal is to use these regressors to estimate a score that would rank the documents in the correct order. Note that since the relevance labels are already given to you in the test set, it is possible for you to compute nDCG scores for the rankings generated. You can use the validation set to tune model (hyper-)parameters. **However, you must refrain from using test set for any form of model tuning**. It is important to note that the hyper-parameters you choose must be common across all folds.

1.3 Metrics

You are required to report the following metrics:

1. nDCG@{5,10,100}
2. Training time
3. Test time (for generating ranking for all queries)

1.4 Report

You should submit a detailed report on how you went about in terms of

(i) Modeling of the problem including loss functions,

(ii) Hyperparameter selection and how you went about making optimal choices,

(iii) Performance on the training, validation and test portions of each fold,

(iv) Observed runtimes for training and test,

(v) Choices you made to improve the runtime.

and any other detail / findings from your experiments to improve the performance of rankings you generate.

1.5 Submissions

Apart from the report (see Section 1.4 above), you must submit your implementation of each of these models. You will have to fix the hyperparameters that you have tuned so that they work on all folds. Only one set of hyperparameters will be accepted.

Program Structure:

1. SVR:
   `SVRLetor.py [fold-directory] [output-file]`
   where, `[fold-directory]` is the full path of the directory containing the three files (training, test, and validation), and `[output-file]` is the rankings generated are printed in order (best ranked first, in non-increasing order) in the `trec_eval` format used in the Assignment 1. **Do not print the features.**

2. GradientBoostedRegressor:
   `GBDT.py [fold-directory] [output-file]`
   where the arguments hold the same meaning as above.

3. MLPRegressor:
   `MLP.py [fold-directory] [output-file]`
   where the arguments hold the same meaning as above.

Result Output Format:

We will adopt the format used by the **trec_eval** (`https://github.com/usnistgov/trec_eval`) tool for verifying the correctness of the output using a program. Therefore, it is **very** important that you follow the format exactly as specified below. The result file format instructions are given below:

Output Format

Lines of results_file are of the form

qid	**iteration**	**docid**	**relevancy**
Q	0	SZF08-175-870	1.0

giving document id (a string, as given in the collection) retrieved by query qid ('Q') with relevancy (i.e., the score as a floating point number generated by your regressor). It is expected that for each qid, the docid's are sorted in non-increasing order of their relevancy values (typically the scores estimated by the regressor). The result file may not contain NULL characters.

1.6 Tentative breakup of marks assignment

The break up of the marks:

SVR	10
Gradient Boosted Regressor	10
MLP Regressor	10
Report	30
Total	60

2 Bonus Part

In this assignment, you have a bonus part worth **15 marks** (in addition to the marks above). In this part, you are given a much larger dataset with significantly larger total number of features, over which you have to experiment.

Dataset - Learning to Rank: This dataset is composed of a set of queries, a set of triples of (query, document, grade), where each query is represented by an identifier, document with a set of features drawn from a universe of about 700 features[1], and a grade from the ordinal set {4, 3, 2, 1, 0} –corresponding to {perfect, excellent, good, fair, bad} respectively.
The dataset is split into training, validation and test set with the following sizes:

- bonus.train.txt – 473, 134 entries
- bonus.valid.txt – 71, 083 entries
- bonus.test.txt – 165, 660 entries (**Will *not* be released**)

Note that there are no explicit document identifiers given.

Average number of documents per query is about 24, although some queries could have much larger than 100 documents.

Data format is similar to the TD2003 and TD2004 datasets with only one variation – there are no document identifiers at the end of the line. So you can consider two documents to be same if their feature sets are same. Otherwise, all documents are distinct (if required).

You are free to conduct any kind of optimizations needed for efficiency and effectiveness of the metrics, including but not limited to – feature selection, feature engineering, dimensionality reduction, etc. Note that you can only tune your method based only on the validation set you are given.

2.1 Bonus - Submission format

If you are attempting the bonus part, you must include it in the same report under a section titled **BONUS TASK**.

The files you need to submit for the bonus part follow the same program and argument convention as the main part *except you need to prefix the files with "bonus_"*. That is, the three files you submit for bonus part will be named as `bonus_SVRLetor.py`, `bonus_GBDT.py`, and `bonus_MLP.py` respectively.

[1] Note that not all documents will have all the 700 features. If a feature is missing, you can consider its value to be 0.