Data Mining: Learning from Large Data Sets - Spring Semester 2014
Instructor: Prof. Andreas Krause

Course Project
Part 4

Explore-Exploit Tradeoffs in Recommender Systems

1 Introduction

The goal of this task is to learn a policy that explores and exploits among available choices in order to learn user preferences and recommend news articles to users. We use real world log data shared by Yahoo!. The original data was collected over a 10 day period and consists of 45 million log lines that capture the interaction between user visits and 271 news articles, one of which was randomly displayed for every user visit. Note that not all 271 articles are available in every round. Instead, we have a pool of roughly 20 articles to choose from in each round.

In each round, you are given a user context consisting of a timestamp and a six dimensional feature vector and a list of available articles for recommendation. Your task is to then select one article from this pool. If the article you selected matches the one displayed by the random policy, then your policy is evaluated for that log line. Otherwise, the line is discarded. Since the articles were displayed uniformly at random during the data collection phase, there is roughly 5% chance that a given line is evaluated.

2 Dataset description

The log lines given in *log.txt* are purely for testing the syntax and expected behavior of your code, and is not representative of the actual values in the real log. Use this file only to check that you are parsing the input correctly.

Every call to your policy contains the following parameters:

- Timestamp: long int.
- User features: 6 dimensional double vector with the first dimension always set to a constant 1.
- Available articles: list of the article IDs that are available for selection.

In addition, you are given access to a separate file (*articles.txt*) that consists of the six dimensional feature vector of every article. For the testing phase, this file is also replaced by a random features file which has the same structure as the actual features file (although the number of lines in the file might vary). Each line is formatted as follows:

- ArticleID feats[0] feats[1] feats[2] feats[3] feats[4] feats[5]
- article ID is a long int, feats contains doubles.
- feats[0] is always 1.

3 Code

In the handout for this project, you will find the 1000 log lines (*log.txt*), and two Python code files (*policy.py* and *evaluator.py*). Your task is to complete the functions `recommend` and `update` in *policy.py*.

The provided evaluator will call your `recommend` function for every line in *log.txt* and will call your `update` function with the following argument:

- 1: The article you recommended was the one chosen by the policy, the user clicked on it.
- 0: The article you recommended was the one chosen by the policy, the user did not click on it.
- -1: The article you recommended was not the one chosen by the policy.

Please ensure that your method always returns an article from the given list. Failing to do so will result in an exception and the execution will be aborted.

4 Evaluation and Grading

Upon submission, we will run your algorithm on the full log file and the real article features will be provided.

Your task is to maximize the fraction of times the article chosen by your policy is clicked by the user.

We will compare the final score of your submission to two baseline solutions: a weak one (called "baseline easy") and a strong one (called "baseline hard"). Both baselines will appear in the rankings together with the score of your submitted predictions.

Performing better than the weak baseline will give you 50% of the grade, and matching or exceeding the strong baseline on the **test set** will give you 100% of the grade. This allows you to check if you are getting at least 50% of the grade by looking at the ranking. If your score is in between the baselines, the grade is computed as:

$$\text{Grade} = \left(\frac{\text{CTR}_{BH} - \text{CTR}}{\text{CTR}_{BH} - \text{CTR}_{BE}}\right) \times 50\% + 50\%$$

4.1 Time limits and Submission limits

Each submission will be given maximum of 60 minutes to execute on the entire log. Each team will be allowed a maximum of 30 submissions. Memory limit is 2 GB. Make sure that you don't submit your solutions in the last moment since you will not be able to get the feedback in time.

4.2 Report

At the end of the course you will be requested to provide a team report which describes your solution for all project tasks. You should update the template given with the first task and add a section titled "Explore-Exploit Tradeoffs in Recommender Systems". The maximum length of a task report is 2 pages (which means that the final report will contain a maximum of 8 pages). You will be required to upload the reports **after the last project of the course.**

4.3 Deadline

The submission system will be open from **Wednesday, 14.05.2014, 17:00** until **Wednesday, 04.06.2014, 23:59:59**.

5 Software

The following is available on each machine in the cluster:

- Python 2.7
- NumPy 1.8.0
- SciPy 0.13.1
- scikit-learn: 0.14.1