Social Choice – Programming Assignment 2 (HW3)

In this exercise you will give a shot at truth discovery by implementing some algorithms learnt in class. In addition, try a truth-discovery algorithm of your own. Deadline is 20/1/2020. 2 points are deducted per day for late submission.

Scenario:

A city planner tried to measure the heights of 25 buildings. In order to do that, he invited some of the finest architects he knows to estimate the buildings' heights by looking from his office's window. Each architect ranked the 25 buildings from lowest to highest and submitted his rankings to the city planner. He, however, had no idea how to derive the real rankings of the buildings from those estimations and contacted you for help.

The city planner has some information regarding the true rankings of some of the buildings. This information is verified and he hands it to you so you can use it for an evaluation of your algorithms.

Your assignment is to write a Python program according to the following demands:

1. Implement functions that calculate PTD, DTD, unweighted (UW). Each of the first three functions should have two versions: one for Borda, and one for Copeland. The input to these functions is only the preferences S and the output is the estimated ranking $\hat{x}$ and voters' fault levels $\hat{f}$ (See algorithms below).
2. Create two scatter plot graphs (one for Borda and one for Copeland) that show, for each architect, the distance from the truth $d(s_i, x^*)$ vs. the proxy distance π_i. Each architect is a point on the graph. The distance function throughout the exercise is Kendall-Tau and the truth is the true rankings you are given **over 13 buildings.**
    - Note that you only need to consider how i ranked these 13 buildings. It is advised that you create a smaller input file S' that only has these 13 buildings.
3. Create two graphs (one for Borda and one for Copeland) that show the average error as a function of sample size. Sample sizes are in the range (5, 85) with intervals of 5. The sample size means that you randomly take only part of the votes (architects) and run the algorithms according to them. For each sample size, calculate the average true error over at least 500 random samples from S'. Each graph should have 3 curves- for DTD, PTD and unweighted. The true error is the distance $d(\hat{x}, x^*)$ **over the 13 buildings**.
4. Your program should read the votes S, call the 6 functions mentioned in section 1, and create a csv file with your estimations $\hat{x}$ **over all 25 buildings**. The csv file should have 6 rows as follows:
    1. DTD Borda
    2. DTD Copeland
    3. PTD Borda
    4. PTD Copeland
    5. UW Borda
    6. UW Copeland
5. (Bonus – up to 15 points based on originality and performance on all 25 buildings): Implement your own truth-discovery method.
    - Submit the code
    - add its output to the graphs in task 3
    - add your estimated ranking over all 25 buildings as a 7th row in task 4

The algorithms (based on Truth Discovery via Proxy Voting / Meir, Amir, Cohensius, Ben-Porat and Xia)

ALGORITHM 5: DISTANCE-FROM-OUTCOME-BASED-TRUTH-DISCOVERY (D-TD)

Input: Dataset $S \in X^n$.
Output: $\hat{f}^0 \in \mathbb{R}^n; \hat{x} \in \mathcal{X}$.
Set $\boldsymbol{y} \leftarrow \mathrm{g}(S)$;
for *each worker* $i \in N$ **do**
  Set $\hat{f}_i^0 \leftarrow d(\boldsymbol{y}, \boldsymbol{s}_i)$;
  Set $w_i \leftarrow SetWeight(\hat{f}_i^0)$;
end
Set $\hat{x} \leftarrow \mathrm{g}(S; \boldsymbol{w})$;

ALGORITHM 1: PROXY-TRUTH-DISCOVERY (P-TD)

Input: Dataset $S \in \mathcal{X}^n$.
Output: Estimated fault levels $\hat{\boldsymbol{f}} \in \mathbb{R}^n$; answers $\hat{x} \in \mathcal{X}$.
Compute $d_{ii'} \leftarrow d(\boldsymbol{s}_i, \boldsymbol{s}_{i'})$ for every pair of workers;
for *each worker* $i \in N$ **do**
  Set $\pi_i \leftarrow \frac{1}{n-1} \sum_{i' \neq i} d_{ii'}$;
  Set $\hat{f}_i \leftarrow EstimateError(\pi_i)$;
  Set $w_i \leftarrow SetWeight(\hat{f}_i)$;
end
Set $\hat{x} \leftarrow \mathrm{g}(S; \boldsymbol{w})$;

- X is the set of all possible rankings.
- EstimateFault() and SetWeight() functions are defined as we saw in class for binary questions.
- $g()$ is the (weighted) voting function
- UW(S) simply returns $g(S)$

The Data:

Voters.csv: the estimations (rankings) of 85 architects (denoted by $S = (s_1, \dots, s_n)$.

Votes – Truth.csv: the true rankings among 13 of the 25 buildings. The first row in this file is the buildings for which the truth is supplied, and the second row is the true order among them (denoted by x^*).

Notes:

- All ties should be broken uniformly at random
- When implementing PTD, DTD you need to use $w_i = \log\left(\frac{1-f_i}{f_i}\right)$ as the weight function. Since this function is not defined for some values, assign a very small or very large (up to 1) value for w_i if you encounter such case. The small value should be selected s.t it is smaller than any other w_i and the large value should be selected similarly. Pay attention to which cases demand a small value and which require a large value.
- For PTD, use $\mu = 0$.
- Unweighted- simply running the methods. For example, Borda UW is the Borda rankings.
- Your program should read the votes files attached to the exercise- it will be also tested on different votes files. Specifically, your program should assume the votes file is in the same directory and read it from "votes.csv" (and not as input to the program), and create the csv file, to be named "estimations.csv", into the same directory.
- A template is attached to the assignment- feel free to use it as a base for your code as it meets the requirements; however, you are not required to do that.

Submission instructions:

- A short PDF file that contains an explanation of your algorithm (If you chose to create one) and the graphs mentioned in sections 2, 3.
- Your code (a .py file) that creates the required csv file.
- Both files should be submitted as a .zip file.
- Submission is in pairs.
- Make sure that the ID's of both submitters are either the .zip file's name (separated by _), or mentioned in the PDF file.