Progress report - Week 1 - 9/02 - 9/09

Parul Gupta (pargupta@umass.edu)

September 8, 2020

1 Milestones Planned

The following milestone was planned for this week:

1. Read Science paper 'Preventing undesirable behavior of intelligent machines'[1].
2. Implement Seldonian algorithm from aisafety.cs.umass.edu

2 Milestones Achieved

2.1 Paper reading: Preventing undesirable behavior of intelligent machines

STATUS: Done
Things done here:

1. Read the Science paper thoroughly.
2. Watched the videos on aisafety.cs.umass.edu that relates to this topic.
3. Browsed through the supplementary material of the paper.

Brief summary/things learnt:
Seldonian algorithm is a framework tackles the problem of regulating the undesirable behavior of machine learning algorithms on various real world data and transfers the responsibility of the machine learning to be 'fair' or 'safe' from user of the algorithm to designer of the algorithm (i.e. at the creation step itself by ML researcher).

The researcher provides an interface to the user which takes input as the data and the behavioral constraint (g). The data is then split into 2 parts - candidate data and safety data. The behavioral constraint and candidate data are passed through candidate selection process to get candidate solution, θ_c. Safety test is done on safety data using this θ_c. If it is passed, we return θ_c, else we return 'no solution found' (NSF). Figure 1 from the paper shows this process.

I went through both regression and classification problem on GPA prediction

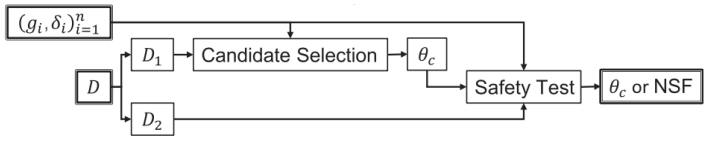


Figure 1: Overview of Seldonian Algorithm [1]

and also the RL problem for diabetes to understand the real life applications of seldonian for safety and fairness. I also went through the supplementary material to understand Seldonian Optimization Problem and Quasi-Seldonian algorithm. As I want to work on fairness track, I studied the section on the fairness in the supplementary material - fairness for supervised learning, fairness constraints and its applications in real world.

2.2 Implementation of Seldonian algorithm

STATUS: Done
Things done here:

1. Went through the complete tutorial on Seldonian algorithms.
2. Fetched the Python code from the website.
3. Setup the environment for the code.
4. Went through the code to understand the working of the *candidateSelection* and $safetyTest$. Executed the implementation and compared the results with the ones present provided on the website.

Brief summary/results:
The tutorial was smooth to follow through and I was able to get things to running.
Sample Experiment: The output of the sample experiment matched the solution on the website:

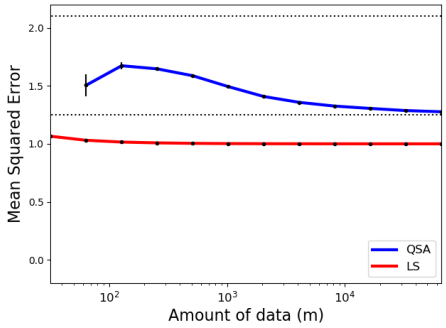
```
parulgupta$ python3 main.py
A solution was found: [0.5844721756, 1.0560192943]
fHat of solution (computed over all data, D): -1.349482921456559
```

I ran the Python tutorial code with the following parameters:

1. Workers = 4.
2. Trials = 70.
3. Delta = 0.1.

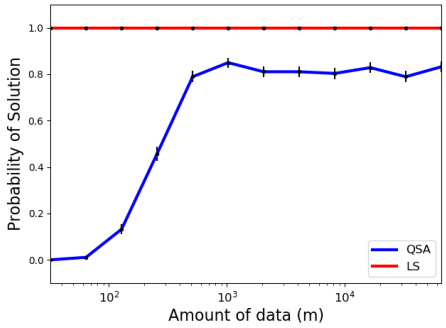
In the below graphs, LS = least square linear regression and QSA = Quasi-Seldonian linear regression algorithm.

MSE Plot: It produced the following graphs which was similar to the graphs given in the website-



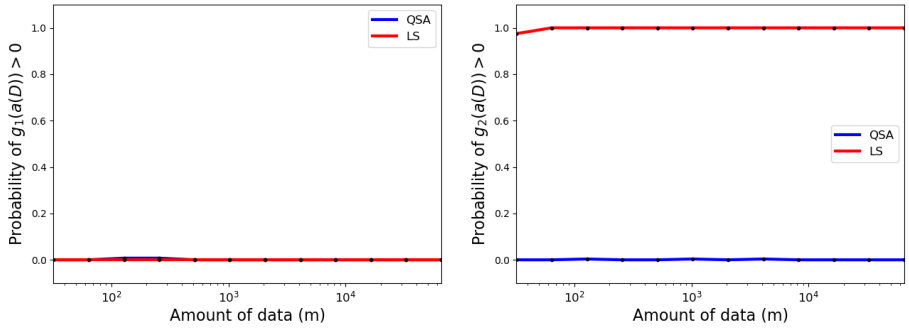
Dotted lines to indicate the desired range of MSE. All the obervations mentioned in the tutorial website match. No solution was found for less amount of data. We see a convergence towards lower bound of desirable range with more data in case of QSA (as expected).

Probability that solution was returned: Following is the graph for probability that each method returned a solution-



In case of LS, solution is always returned. However, for QSA, it stabilizes after a point at around 0.8.

Probability of undesirable behavior: Following are the graphs for probability that each algorithm produced undesirable behavior-



On analyzing the graphs, we see that the QSA line is not exactly straight flat line at Probability=0. It does move up at times; however, it stays below the

delta of 0.1.
Overall, the setup is complete now and all the results came as expected.
I have setup a private github repo where I intend to put all the code: fair-work

References

[1] Philip S Thomas, Bruno Castro da Silva, Andrew G Barto, Stephen Giguere, Yuriy Brun, and Emma Brunskill. Preventing undesirable behavior of intelligent machines. *Science*, 366(6468):999–1004, 2019.