Seldonian Algorithms Library Project report

Sahil Girish Yerawar

Overview

Seldonian algorithms are a class of algorithms in the space of parametric machine learning and Reinforcement Learning, which help in enforcing the quality of fairness in the performance of these models. With the advent of massive use of Machine Learning and Reinforcement Learning, introducing some fairness or safety constraints, apart from curating the dataset, is the responsibility of the Machine Learning (ML) practitioners who want to deploy their models for public use. In this report, we work with the Seldonian Toolkit [1], which is a library tool helping researchers, ML practitioners and data scientists to easily enforce fairness constraints to their models.

How Seldonian Algorithm works

For a given Seldonian algorithm to operate, the dataset D is divided into two parts D_{cand} and D_{safety}. The D_{cand} portion is used in the process of *Candidate Selection*, which returns the candidate solution θ_c. This θ_c is supposed to be the best model which performs the best on D_{cand} with no constraints. This model is then passed on to the *safety test*, which uses D_{safety} and θ_c to validate that there is enough confidence that θ_c will satisfy all the constraints, and therefore safe to pass forward. By this method, the Seldonian algorithm returns the best model which ensures fairness in its prediction without significantly affecting its performance.

Application

We utilize the Seldonian algorithms through the Seldonian Toolkit library [1] (link) to develop models which can assign tutorials to students with a certain degree of fairness. We utilize the Intelligent Tutoring Dataset which was specially curated for the Robinhood algorithm [2], to develop a simple environment which can simulate the dataset, and then apply Seldonian algorithms to analyze its effeciency for this use case.

Data Analysis

The first step of this process is to understand the dataset and its components. It consists of around 2600 people, with about 1403 of them as males and 1178 of them as females. Each person has the chance of getting one out of three tutorials, and on completion of tutorials, they get an integer score between 0 and 10 (both inclusive). As such, we model this problem as a contextual bandit problem, with the state being the gender(Female/Male), the action being assigning tutorial 1,2 or 3 and the reward being the score obtained by that person on the tutorial.

The only stochastic factor that is worth considering in this dataset, is the reward distribution. For each gender and tutorial tuple, there exists a score distribution between 0 and 10. We model frequency of the occurence of a score for each gender and tutorial pair as a probability and use it to build a reward distribution $R(s, a)$.

Model Creation

With the help of Seldonian toolkit [1], we can create a separate environment which can model the behaviour of the Intelligent Tutoring Dataset by estimating the reward probabilities from it. The goal of this model is to find policies which are better than the uniform random policy. So, the model constraint becomes:

$$G : J_{\pi}^{new} \geq J_{random}$$

We utilize this Seldonian library for our use case by creating a RL environment for our dataset, by creating a new class derived from `RL.environments.Environment` class provided by the Seldonian toolkit [1]. It is in this model that we encode the model transition function and the reward function in the form of their probability distributions. The code for this environment is present in `Environment/RL_environment.py` module.

After the formation of the RL environment dataset, we randomly sample 10000 episode of tutorial assignment for both the genders, and create the dataset for the further downstream use.

We compute the mean of rewards obtained for all the 10000 episodes in total, which is used as a baseline in our safety constraint. For reference purpose, we got this value as 2.9743, which made the constraint objective:

$$G : 2.9743 - J_{pi} \leq 0$$

Essentially, this constraint helps in finding better policies which can beat the random policy performance. Equipped with this objective, we generate the `RLSpec` object for this sample of episodes. The spec object generated captures all the information of the RL agent, environment and the safety constraint imposed. We use this spec object in the main `SeldonianAlgorithms` object to obtain those policies which have a good balance of performance while also respecting the safety constraints. The hyper-parameters used in this object are $num\_iters = 100, alpha\_theta = 0.05, alpha\_lamb = 0.01$. The code for this is present in `Robinhood_SA.py`.

Model Experiments

In order to analyze the performance of these algorithms, there is an associated library called `Seldonian Experiments` which have essential utility functions required to create successful Seldonian plots. These plots measure the performance, probability for finding a solution given the constraints and the probability of violating the constraint as a function of size of the training set. The code for this is present in `Robinhood_SA_experiments.py`.

Results

After some hyperparameter tuning related to the Seldonian toolkit [1], the Seldonian algorithms do work on this environment. Figure 2 show the performance, solution rate and the failure rate of this experiment. In particular, this figure shows that the new algorithm can achieve a better return (around 4.2) with about 95% confidence without violating the constraint for any size of the dataset.

Since these graphs are generated from additional simulations based on the `spec` object created, the `SeldonianAlgorithms` Library can also create the gradient descent plots based on the logs provided by the `SeldonianAlgorithms.Run` method. The plots are shown by Figure 1. In Figure 1, the right-most plot shows us, that as the iteration progresses, the IS estimate graph becomes a plateau around -4.2 which is in confirmation with the additional simulations done by the `Experiments module`.

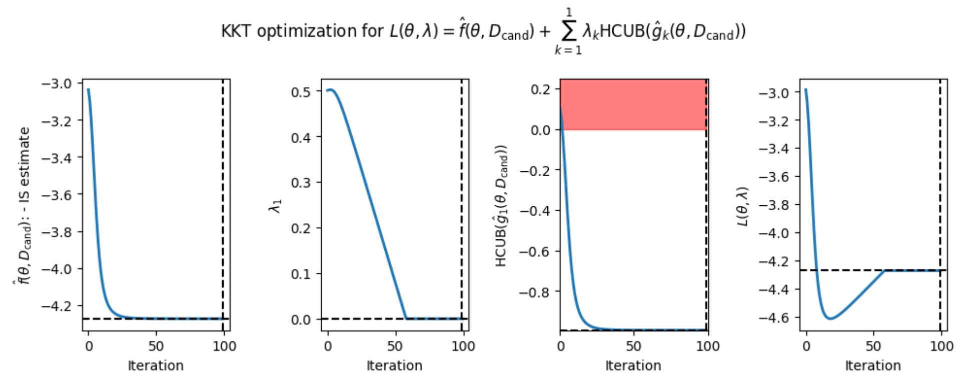

Figure 1: Gradient Descent Plot of the Seldonian Algorithm Run

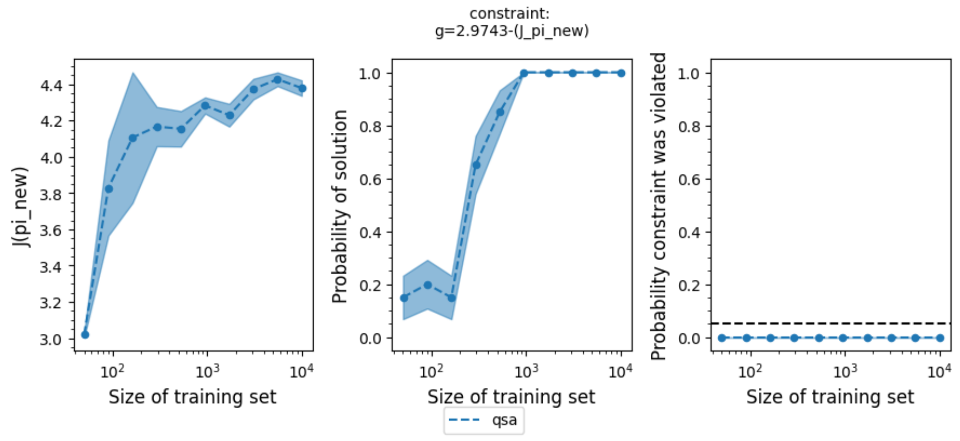

Figure 2: Performance, Solution Rate, and failure rate of the Seldonian algorithm on the future data

Future Work

The next steps of the project are to enforce the gender based constraints mentioned in Robinhood [2], in this environment using the Seldonian toolkit [1]. Currently, these constraints differ based on certain features of the datapoints, like gender, and have different constraints for different genders. This type of constraint handling is not present in the Seldonian Toolkit [1], as of this moment. Hence, this would require some discussions with the developers behind the library to incorporate these features. The eventual goal of this project is to develop high-quality tutorial of applying Seldonian algorithms on the environment denoted by this dataset, to further educate aspiring machine learning enthusiasts and researchers about the extent to which these algorithms are applicable. An alternative goal, which could be worth pursuing, is to further introduce complexities in this dataset, like providing a sequence of tutorial inorder to make the environment of this dataset more like a MDP.

References

[1] S. Algorithms. (2022) Seldonian algorithms reference. [Online]. Available: https://seldonian.cs.umass.edu/Tutorials/

[2] B. Metevier, S. Giguere, S. Brockman, A. Kobren, Y. Brun, E. Brunskill, and P. S. Thomas, "Offline Contextual Bandits with High Probability Fairness Guarantees," in *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems (NeurIPS), Advances in Neural Information Processing Systems 32*, Vancouver, BC, Canada, December 2019,

pp. 14 893–14 904. [Online]. Available: http://papers.neurips.cc/paper/9630-offline-contextual-bandits-with-high-probability-fairness-guarantees