SDE project: Test Case Prioritization

Nikhil More (B18CSE037)
Saurabh Shashikant Burewar (B18CSE050)

Introduction

Recently, Reinforcement Learning (RL) has shown great potential in various challenging scenarios that require continuous adaptation, such as game playing, real-time ads bidding, and recommender systems. Inspired by this line of work and building on initial efforts in supporting test case prioritization with RL techniques, this paper implements algorithms for RL-based test case prioritization in a CI context. To this end, taking test case prioritization as a ranking problem, the paper models the sequential interactions between the CI environment and a test case prioritization agent as an RL problem, using three alternative ranking models.

Objective - The objective of this project is to implement the RL algorithms evaluated in this paper [1] for training the RL agent for test case prioritization.

Summary - We have implemented the pointwise ranking approach for this problem and compared the performance with what is discussed in the paper. The performance of the listwise approach is the worst while the pairwise approach in the paper is said to be the best among the three ranking models. The pointwise approach, which we have successfully implemented gives almost similar performance as in the paper, however it is not the best model in the paper, hence the performance of our implementation is not equivalent to the performance given in the paper. The datasets used in the paper are publicly available, so we have used the same for our experiments.

Comparison with existing work

The existing paper on this gives an extensive approach to the test case prioritization using three ranking models - pairwise, pointwise and listwise. It also uses different RL algorithms for each ranking model like DQN, PPO, A2C, etc.

In our implementation, we have built the pointwise ranking model for test case prioritization, as introduced in the paper. This model uses 5 different RL algorithms including PPO2, A2C, TRPO, TD3, ACKTR.

Considering only the structural features of the test cases can ignore the test cases who rank lower in structural features but are failing continuously in testing. The test cases which are failing should be given more priority over others and hence considering other features of the test case is important. The paper has made the same attempt of including extra features of the test cases so as to improve ranking prediction. These features include average time, last execution time, failure history, age (number of times failed), etc. These features are included to consider the execution history of the test cases along with their structural features. The test cases with failure in the last cycle are given priority over the test cases which passed in the last cycle. The test cases then are sorted in ascending order according to their execution time in the last cycle.

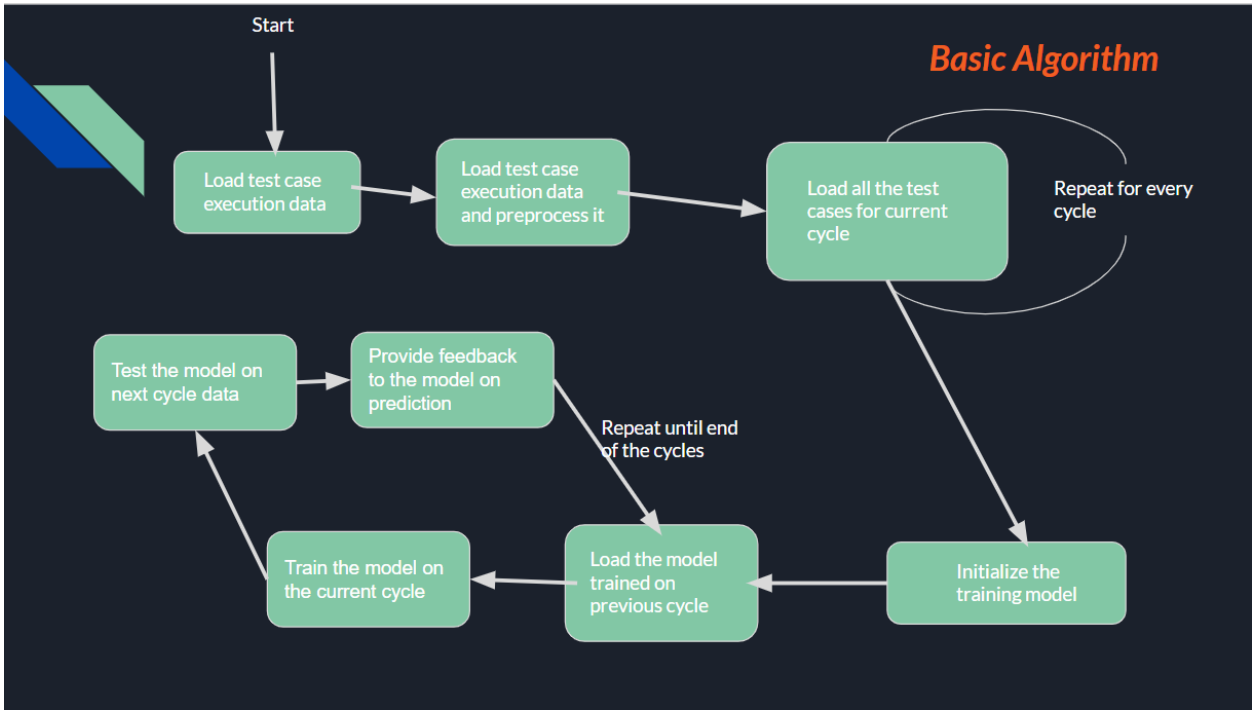
A pointwise ranking approach takes the features of a single test case and uses a prediction model to provide a relevance score for this test case.
The authors of the paper have also used pairwise and listwise ranking models where they have considered extra features apart from the ones above such as time and duration groups of the test cases. But as we are considering one test case at a time in the point wise model, the features like time and duration groups do not contribute to the ranking and hence were discarded in the implementation.

We got the following results from our implementation and given the corresponding results given by the actual implementation in the paper, it seems to be performing nearly equivalent for some algorithms while it is worse for others. But given the margin of error for our implementation, the performance seems to be slightly worse than the actual implementation given in the research paper.

Results :

Algorithm	**Paper results (PO)**	**Our results (APFD, randomAPFD, optimalAPFD)**
A2C	.57±.23	**0.5548**; 0.5630; 0.9817
TRPO	.57±.23	**0.4857**; 0.5101; 0.9817
PPO2	.57±.24	**0.3922**; 0.3800; 0.9817
TD3	.58±.24	**0.4207**; 0.3434; 0.9817
ACKTR	.57±.24	**0.4207**; 0.5589; 0.9817

Schematic diagram



Code

The code can be found in the link below -
https://github.com/saurabhburewar/ML_RL-test-case-prioritization

Report

This report can also be accessed through the link below -
https://docs.google.com/document/d/1gFGQ7mrjcwDdhtMW6piexYHsCF_5sitC-3PUV_pV-F0/edit?usp=sharing

Research Paper

The research paper that was implemented in this project can be found below -
https://arxiv.org/pdf/2011.01834.pdf

Demonstration

The video demonstration can be found below -
https://youtu.be/UMqSxRbUb3Y