REPRODUCIBILITY CHALLENGE: THE IN-SAMPLE SOFTMAX FOR OFFLINE REINFORCEMENT LEARNING

Jack Hilton-Jones & Sam Hilton-Jones
Department of Electronics and Computer Science
University of Southampton
Southampton, SO17 1BJ, UK
{jhj1g23,shj1g20}@soton.ac.uk

ABSTRACT

Within this report, we reproduce key details of the ICLR 2023 notable top 25% paper: 'The In-Sample Softmax for Offline Reinforcement Learning' (Xiao et al., 2023). We recreate the tabular environment and reproduce the performance graphs on a D4RL environment by training the agent offline and then fine-tuning using online learning, concluding that this paper is largely reproducible with minor inconsistencies.

1 INTRODUCTION

The paper chosen for this Reproducibility Challenge is 'The In-Sample Softmax for Offline Reinforcement Learning' (Xiao et al., 2023). This paper focuses on challenges faced within offline Reinforcement Learning (RL). The primary challenge addressed is how the agent must learn policies from limited data, which may be skewed or suboptimal, without further interaction with the environment. Bootstrapping is an issue when using offline RL methods, as this estimates the future values based on the current values. This can be problematic if the value is an overestimate, which is likely to occur when there are actions that are never sampled in a state. A strategy to overcome this issue is to consider bootstrapping for a limited amount of actions, defined here as taking the *in-sample max*. Xiao et al. (2023) proposes a new approach that uses an in-sample softmax in an entropy-regularised setting while maintaining theoretical properties such as contraction and convergence. The paper can show that decreasing the temperature parameter allows the in-sample softmax to approximate the in-sample max, enabling more accurate value estimations. This improvement helps mitigate issues of overestimation and instability that occur when using experience with poor action coverage, leading to better policy performance and stable convergence. This uses an in-sample actor-critic (InAC) algorithm that allows updates based primarily on the dataset, which avoids the need to estimate the policy under which the data was collected. The paper claims that using this algorithm performs just as well or outperforms baseline algorithms with a simpler approach and without additional fine-tuning.

2 TARGET QUESTIONS

We aim to reproduce the performance of this algorithm on all three main aspects of the paper; a tabular environment, a more complex continuous action space using D4RL (Fu et al., 2021) and complete both offline learning and online fine-tuning of this algorithm to optimise performance. This provides us with the following target questions to investigate:

- Are sufficient details provided to reproduce the results on the tabular environment and does the agent code adapt to a newly created environment and dataset?
- Using a D4RL environment, is the agent able to achieve similar performance with the InAC algorithm trained offline to that in the paper?
- How can we fine-tune the offline model and apply online learning for continuous interactions with the environment to achieve a higher performance?

3 TABULAR ENVIRONMENT REPRODUCTION

Within this paper, the in-sample softmax is first tested on a 'sanity-check' environment. This is used to ensure that the method converges to an optimal policy found by an oracle method that exactly eliminates out-of-distribution (OOD) actions when bootstrapping. The code for this environment was both incomplete and unusable with the InAC algorithm. Therefore, we decided to create our own implementation from scratch, both creating a Grid World environment and generating the experience datasets for the agent to sample from. To ensure reproducibility, we abandon the partial tabular code, formulating the environment and datasets with the same properties using a new code format.

The paper describes four datasets to train the agent; expert, random, mixed and missing-action. The expert dataset uses a value iteration function, leading to highly optimal experiences with low action-state coverage. The random dataset produces many sub-optimal actions but has good action-state coverage. The mixed dataset combines the random and expert datasets, with a 0.99:0.01 split. Finally, the missing-action dataset, within the main section of the paper, is described as 'removing all transitions taking down actions in the upper-left room from the mixed dataset'. However, the Appendix describes this as 'constructed by all going-down transitions in the upper-left room from the mixed dataset'. We believe, based on these differing descriptions and observing the figures provided in the paper, that there is a discrepancy between these datasets that is not clarified by the authors.

We generate these datasets for 10,000 transitions, where we name the description for the missing-action dataset given in the main part of the paper as Missing-Action 1, and for the description in the Appendix, Missing-Action 2. We then use the agent, which we have verified matches the specification given in the paper, on our environment and datasets for 10,000 episodes, where there are 100 steps per episode. We recreated the reward function described such that the agent receives a reward of 1 for every timestep it is positioned in the goal coordinates and 0 everywhere else. We also changed the reward function used to a Manhattan distance-based measure and found convergence remained the same, therefore we test using only the original function. We averaged the return per episode over the five runs for each of the datasets, however, for the random dataset, two of the runs produced poor convergence and therefore significantly reduced the average return. Hence, we choose to plot the moving average with a window size of 50 for the five runs for each dataset. However, for the random dataset, we excluded the two runs with poor convergence, we plot this in Figure 1.

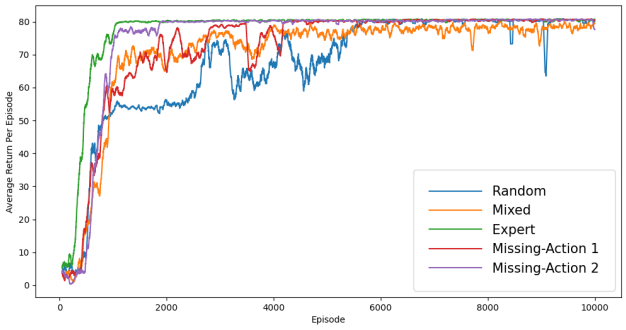


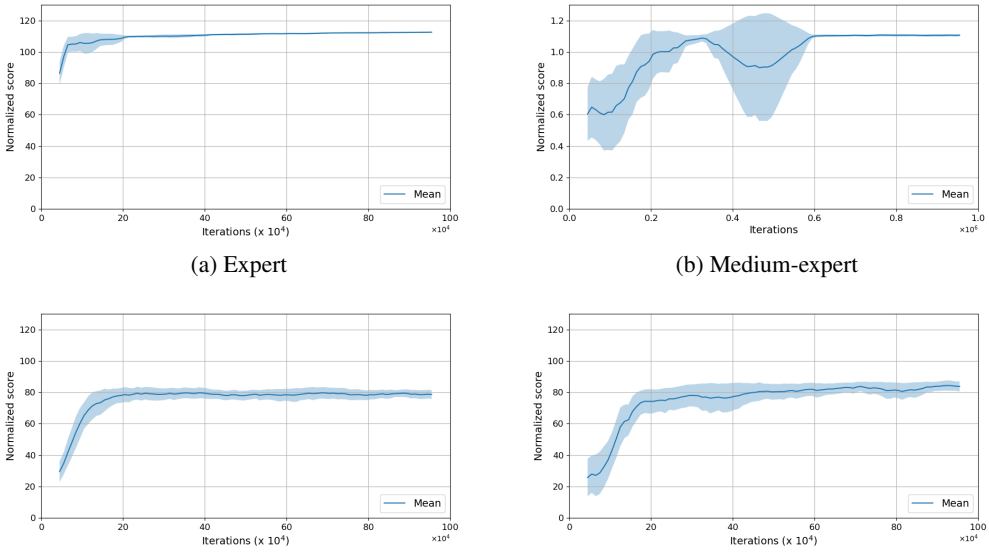
Figure 1: Performance of the In-Sample Softmax method on different datasets with varying action-state coverage within the tabular environment over 10,000 episodes with 100 steps per episode.

We can now compare these results to those provided in the paper. The paper used a parameter search on the learning rates, however, the learning rate used for the production of the figures is not specified. Therefore, we choose to sample these learning rates by categorising them into a comparatively large learning rate, 1×10^{-2}, medium learning rate, 3×10^{-3} and small learning rate, 1×10^{-3}. When completing this with the expert dataset, we find that the large learning rate resulted in no convergence while the medium and small learning rates produces good performance. However, when testing with the random dataset, the medium learning rate does not show any convergence, hence we choose to

use a fixed learning rate of 1×10^{-3} for all datasets as this is the only value that produces good convergence. Because of this, we expect many of the discrepancies in results to be a consequence of this change in the learning rate. Within the paper, the agent has rapid convergence with good stability for all datasets. However, in our results, while the agent converges to the same value for all examples, including our new dataset Missing-Action 2, the convergence rate is much slower. In particular, the agent trained on the random dataset included both the two poor runs and the slowest convergence rate indicating it struggles to efficiently learn an optimal policy quickly due to the broad, and suboptimal nature of the data. Finally, we run a number of unit tests to ensure that the algorithm and environment are interacting correctly and producing coherent learning values.

4 Reproducing Results within D4RL Environment

Using the Walker2D Mujoco environment provided by D4RL, InAC was tested using four datasets, expert, medium-expert, medium-replay and medium. The medium dataset collected by the Soft-Actor-Critic (SAC) halfway through training gives a partial random set of trajectories. This compares to the expert dataset which consists of trajectories collected through a fully trained SAC agent giving near-optimal decisions. Medium-expert combines the expert and medium datasets, providing a balanced set featuring both optimal decisions and varied trajectories. The medium-replay combines the medium dataset with the replay buffer, which is a storage of past training and results that the agent has encountered. This provides a more robust model by allowing the agent to stabilise learning, providing a range of training history. Our results use a learning rate of 3×10^{-4}. This is run for 10,000 episodes using 100 steps averaged over 5 random seeds and plotted after using a smoothing window of size 10 with a 95% confidence interval, this is shown in Figures 2a, 2b, 2c and 2d.



(a) Expert (b) Medium-expert

(c) Medium-replay (d) Medium

Figure 2: Normalized scores for different datasets within the Walker2D environment

We now take the offline-trained deep RL agent and fine-tune it online, this code was omitted from GitHub and therefore we integrate our code into the model to reproduce online learning. Initially, the replay buffer is populated with the parameters of the offline data. The agent is then trained and designed to constantly interact with the environment allowing it to learn from interactions in real time, continuously updating the policy and value functions. The paper did not specify the learning rate used, therefore we analyse the following learning rates for our online learning: 3×10^{-4}, 3×10^{-5}, 1×10^{-5}, 3×10^{-6}. Here we found fine-tuning with a learning rate of 1×10^{-5} was optimal for the expert and medium-expert dataset whereas a learning rate of 3×10^{-4} was optimal for online learning with medium-replay and medium datasets. We expect this outcome as the medium-

expert and expert are higher quality datasets and require careful tuning, whereas medium-replay and medium are noisier, requiring faster corrections and more substantial updates.

5 FINE-TUNING

There is a discrepancy between the offline graphs for the Walker2D datasets and the performance tables shown in the paper. Within the GitHub repository, the authors mention an updated fix for the policy network for continuous control and plot the graphs using this. However, the performance tables do not provide an accurate numerical performance value relating to the graphs, particularly for the medium-replay dataset. Our results show a better performance in the offline training of the agent, we argue that our results reflect the paper's figures more accurately after the fixed policy network. Even though our results are less stable due to the reduced runs over random seed, we outperform the paper's online learning results for expert and medium-expert datasets. However, despite being similar, the paper demonstrates better online learning for medium and medium-replay. We conclude that although our results of the offline learning are different to the paper's results, they are a truer reflection of the fixed policy update by the authors. Likely, our results are not an improvement but instead a reflection of the update, which we cannot be certain about, as the paper did not release updated performance scores.

Walker2D	**Offline**	**FineTune**	**Change**
Our expert	**112.34 (0.49)**	**112.0 (0.58)**	-0.34
Paper's expert	110.60 (0.09)	110.90 (0.03)	0.3
Our medium-expert	**111.4 (0.80)**	**112.33 (0.33)**	0.93
Paper's medium-expert	109.00 (0.10)	112.20 (0.03)	3.2
Our medium-replay	**78.8 (2.23)**	83.0 (2.08)	4.2
Paper's medium-replay	69.80 (0.57)	**95.50 (0.10)**	25.7
Our medium	**84.0 (4.04)**	87.33 (0.33)	3.33
Paper's medium	82.70 (0.16)	**89.70 (0.07)**	7

Table 1: Normalised score performance comparison for Walker2D dataset after 1 million offline iterations and 0.8 million online iterations. The standard error is seen in brackets.

6 CONCLUSION

In this report, we have reproduced all of the key aspects of our chosen paper. We begin by creating a new environment and dataset for the agent to sample from and confirm that, despite the explicit learning rates for the figures being excluded, there is sufficient information and details to produce an agent that has both fast and stable convergence in a simple setting. With the notion that we created the environment and dataset from scratch, we also ensure that this agent code is adaptable and reproducible on a customisable environment, confirming that this method is generalisable. We claim we show a truer reflection in numerical scores for the graphs produced in the paper by offline learning, especially for the medium-replay dataset in the Walker2D environment. We then integrate our code into the algorithm to fine-tune the model using online learning. In conclusion, we have found that this paper is reproducible with updates needed to complete the results for the fixed policy network. We verify the robust performance of the InAC algorithm across our experiments and the advantage of approximating an in-sample max using only actions well-covered by the dataset.

REFERENCES

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2021.

Chenjun Xiao, Han Wang, Yangchen Pan, Adam White, and Martha White. The in-sample softmax for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=u-RuvyDYqCM`.