An empirical study of the naïve REINFORCE algorithm for predictive maintenance of industrial machines

Rajesh Siraskar

June 21, 2023

Abstract

The 14th-century friar, William of Ockham, has been attributed for giving us the "Occam's razor" principle – when there are two competing "theories" predicting the same phenomenon, one should prefer the *simpler* of two.

In this empirical study, we document the performance of a simple, early reinforcement learning algorithm, REINFORCE, implemented for a predictive maintenance problem. We compare a very naive implementation of REINFORCE against the predictions of industry-grade Stable-Baselines3 (SB-3) implementations of three advanced algorithms, namely, Deep Q-Network (DQN), Advantage Actor-Critic (A2C) and Proximal Policy Optimization (PPO). Our broad goal was to understand the performance under various scenarios such as a simulation-based environment, three sets of real tool-wear data, added noise levels, and a random chance of break-down. Model performance was measured by how accurately the predictive maintenance agent suggested tool replacement compared to a deterministic preventive maintenance rule based on the tool-wear threshold.

Our findings indicate that the REINFORCE performs significantly well for this particular problem. Across variants of the environment, the REINFORCE algorithm demonstrated an average F1 performance of 0.836 against 0.383 for A2C, 0.471 for DQN, and 0.402 for PPO. As a measure of stability, the overall standard deviation for REINFORCE was 0.041, while A2C, DQN, and PPO standard deviations were 0.059, 0.029, and 0.070, respectively. Across precision on tool replacement, REINFORCE was better

by 0.354 basis points than the best of advanced algorithms and demonstrated a variance lower by 0.004. While the REINFORCE demonstrated better performance for each variant, it was observed that the training was unstable, occasionally producing poor performance models. On the other hand, the SB-3 implementations training was more stable, almost always producing models with an F1 in the range 0.47-0.50.

1 Method and Inference

- why classifction metrics
- why F1beta

1.1 Method - training and testing

- Training: SB-3 - 10 k eps. 3 times. Average their outputs
- Testing:
  - Avg. over 5 rounds.
  - Each round - avg over 40 test cases x 10 test rounds
  - Total: 40 x 10 x 5 = 2000 cases
  - Avgs over: 10 rounds (of 40 cases each) X 5 rounds of **re-trained** SB-3 agents = 50 rounds

1.2 Inference

- Training: SB-3 is also unstable - show examples of results such as A2C/DQN 0.00
- Training: SB-3 is also unstable - SHOW SB-3 tensorboard plots
- Training: SB-3 is also unstable - EXCEL plots of results over the 10 rounds

2 Introduction

Introduced in 1992, the REINFORCE algorithm is considered as a basic reinforcement learning algorithm. It is a policy-based, on-policy as well as off-policy algorithm, capable of handling both discrete and continuous observation and action domains.

In practice the REINFORCE algorithm is considered as “weak” learner and superseded by several algorithms developed since. Most notably the Q-Learning and its deep-neural network version, the DQN, followed by Actor-Critic and one of the most robust modern day algorithms, the PPO.

Reinforcement Learning in Robotics: A Survey - Jens Kober J. Andrew Bagnell Jan Peters - Initial gradient-based approaches such as finite differences gradients or REINFORCE (Williams, 1992) have been rather slow. The weight perturbation algorithm is related to REINFORCE but can deal with non-Gaussian distributions which significantly improves the signal to noise ratio of the gradient (Roberts et al., 2010). Recent natural policy gradient approaches (Peters and Schaal, 2008c,b) have allowed for faster convergence which may be advantageous for robotics as it reduces the learning time and required real-world interactions.

3 Results

3.1 Detailed metrics

	REINFORCE				A2C				DQN				PPO			
Environment	Prec.	Recall	F1	F0.5	Prec.	Recall	F1	F0.5	Prec.	Recall	F1	F0.5	Prec.	Recall	F1	F0.5
Simulated - No noise	0.999	0.645	0.782	0.898	0.335	0.359	0.344	0.338	0.348	0.597	0.410	0.352	0.392	0.211	0.252	0.303
Simulated - Low noise	0.943	0.954	0.948	0.945	0.409	0.318	0.349	0.379	0.273	0.064	0.076	0.108	0.359	0.173	0.205	0.255
Simulated - High noise	0.889	0.974	0.929	0.904	0.471	0.439	0.443	0.455	0.423	0.408	0.295	0.284	0.402	0.205	0.248	0.307
PHM C01 SS - No noise	0.886	0.978	0.928	0.902	0.294	0.337	0.305	0.296	0.350	0.405	0.291	0.269	0.517	0.494	0.471	0.476
PHM C01 SS - Low noise	0.916	0.893	0.903	0.911	0.526	0.645	0.568	0.540	0.321	0.591	0.404	0.343	0.490	0.415	0.443	0.468
PHM C01 SS - High noise	0.757	0.926	0.831	0.784	0.499	0.632	0.542	0.513	0.399	0.402	0.308	0.292	0.403	0.223	0.270	0.325
PHM C04 SS - No noise	0.865	0.959	0.908	0.881	0.515	0.676	0.575	0.535	0.365	0.497	0.383	0.348	0.431	0.239	0.265	0.311
PHM C04 SS - Low noise	0.722	0.980	0.831	0.762	0.399	0.393	0.391	0.393	0.409	0.589	0.410	0.361	0.438	0.299	0.334	0.377
PHM C04 SS - High noise	0.770	0.809	0.787	0.776	0.375	0.456	0.397	0.381	0.408	0.411	0.296	0.282	0.491	0.324	0.362	0.409
PHM C06 SS - No noise	0.996	0.609	0.751	0.879	0.463	0.454	0.455	0.459	0.538	0.780	0.585	0.523	0.402	0.410	0.374	0.370
PHM C06 SS - Low noise	0.968	0.854	0.905	0.941	0.508	0.615	0.548	0.522	0.395	0.593	0.411	0.362	0.454	0.342	0.367	0.404
PHM C06 SS - High noise	0.699	0.912	0.790	0.732	0.480	0.512	0.466	0.467	0.581	0.499	0.417	0.433	0.424	0.199	0.252	0.314
PHM C01 MS - No noise	0.824	0.895	0.856	0.836	0.444	0.284	0.315	0.358	0.313	0.215	0.165	0.175	0.513	0.347	0.395	0.448
PHM C04 MS - No noise	0.752	0.678	0.709	0.733	0.506	0.326	0.368	0.425	0.588	0.642	0.492	0.486	0.472	0.455	0.444	0.452
PHM C06 MS - No noise	1.000	0.643	0.779	0.896	0.499	0.731	0.575	0.523	0.520	0.239	0.209	0.256	0.509	0.260	0.330	0.409

Table 1: Model performance comparison all variants of the environments.

3.2 Overall summary performance

	Precision		Recall		F1-score		F1-beta score	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
A2C	0.448	0.074	0.478	0.084	0.443	0.071	0.439	0.069
DQN	0.415	0.196	0.462	0.033	0.343	0.038	0.325	0.063
PPO	0.447	0.147	0.306	0.090	0.334	0.093	0.375	0.107
REINFORCE	0.866	0.042	0.847	0.054	0.842	0.043	0.852	0.042

Table 2: Model performance summary - averaged over all environment.

3.3 Simulated environment

	Precision		Recall		F1-score		F1-beta score	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
A2C	0.405	0.079	0.372	0.086	0.379	0.076	0.391	0.076
DQN	0.348	0.217	0.356	0.033	0.260	0.041	0.248	0.068
PPO	0.385	0.175	0.196	0.064	0.235	0.080	0.289	0.110
REINFORCE	0.944	0.029	0.858	0.041	0.886	0.032	0.916	0.030

Table 3: Model performance summary - averaged over simulated environments.

3.4 Real data – simple single-variable environment

	Precision		Recall		F1-score		F1-beta score	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
A2C	0.451	0.064	0.524	0.085	0.472	0.067	0.456	0.063
DQN	0.418	0.172	0.530	0.032	0.389	0.034	0.357	0.055
PPO	0.450	0.146	0.327	0.095	0.349	0.095	0.384	0.106
REINFORCE	0.842	0.043	0.880	0.053	0.848	0.043	0.841	0.042

Table 4: Model performance summary - averaged over PHM-2010 environments with simple single-variable environment.

3.5 Real data – complex multi-variate environment

	Precision		Recall		F1-score		F1-beta score	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
A2C	0.483	0.101	0.447	0.081	0.419	0.075	0.435	0.079
DQN	0.474	0.248	0.365	0.038	0.289	0.049	0.306	0.082
PPO	0.498	0.121	0.354	0.103	0.390	0.101	0.436	0.104
REINFORCE	0.859	0.053	0.739	0.069	0.781	0.055	0.822	0.052

Table 5: Model performance summary - averaged over PHM-2010 environments with complex multi-variate environment.

3.6 Algorithm timelines

- 1947: Monte Carlo Sampling
- 1959: Temporal Difference Learning
- 1989: Q-Learning
- 1992: REINFORCE
- 2013: DQN
- 2016: A3C
- 2017: PPO

4 The REINFORCE algorithm

Three key features of any RL algorithm:

1. Policy: π_θ = Probablities of all actions, given a state. Parameterized by θ
2. Objective function:
$$\max_\theta J(\pi_\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} [R(\tau)] \tag{1}$$
3. Method: Way to udate the parameters = Policy Gradient

4.1 Policy gradient numerical computation

1. Plain vanilla:
$$\nabla \theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} R_t(\tau) \; \nabla_\theta \ln \pi_\theta(a_t|s_t) \right] \tag{2}$$
2. With Monte Carlo sampling and approximation: $\nabla_\theta J(\pi_\theta) \approx [\, \sum_{t=0}^{T} R_t(\tau) \; \nabla_\theta \ln \pi_\theta(a_t|s_t) \,]$
3. With baseline: $\nabla_\theta J(\theta) \approx [\, \sum_{t=0}^{T}(R_t(\tau) - b(s_t)) \; \nabla_\theta \ln \pi_\theta(a_t|s_t) \,]$
4. Where, baseline does not change per time-step, it is for the entire trajectory
5. One baseline option: V^π - leads to Actor-Critic algorithm
6. Simpler option: Average returns over trajectory: $b = \frac{1}{T} \sum_{t=0}^{T} R_t(\tau)$

Algorithm

5 About Stable-Baselines-3

- SB3- paper (Raffin et al., 2021), Raffin et al. (2021)
- sb-3 main doc – (SB3, b)
- sb-3 ppo doc – (SB3, a)

6 Method

We normalize the tool wear and other state features, $x \in [0,\ 1] \subset \mathbb{R}$. This allows for adding white noise of similar magnitudes across experiments of different data-sets

7 Results

8 Discussion

9 Conclusion

References

Stable-baselines3 docs - reliable reinforcement learning implementations, a. URL https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html#how-to-replicate-the-results.

Ppo, b. URL https://stable-baselines3.readthedocs.io/en/master/index.html. Accessed: 2023-05-14.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *J. Mach. Learn. Res.*, 22(1), jan 2021. ISSN 1532-4435.