Exploring Munchausen Reinforcement Learning

Mohamed Amine Ketata
Technical University of Munich
Munich, Germany
mohamedamine.ketata@tum.de

Konstantin Pervunin
Technical University of Munich
Munich, Germany
konstantin.pervunin@tum.de

I. Introduction

Most RL algorithms make use of Temporal Difference (TD) learning, which consists in replacing the unknown true value of a transiting state by its current estimate and use it as a target for learning. The idea of Munchausen Reinforcement Learning (M-RL) [1] is to leverage the current policy to bootstrap RL, by optimizing for the *immediate* reward *augmented* by the scaled log-policy of the agent when using any TD scheme.

II. M-RL for discrete action spaces

A. Key equations

In the first part of the project, we combined the idea of M-RL with the most popular RL agent, the Deep Q-Network (DQN) [2]. M-RL assumes stochastic policies while DQN computes deterministic policies. To address this, we can replace the greedy policy used with DQN with the softmax policy and maximize the entropy of the resulting policy along with the return, that is adopting the viewpoint of maximum entropy RL.

Now, to obtain M-DQN, we just have to add the scaled log-policy to the reward. Let $\alpha \in [0,1]$ be a scaling factor, the regression target of M-DQN is thus

$$\hat{q}_{m-dqn}\left(r_t, s_{t+1}\right) = r_t + \alpha\tau ln\pi_{\bar{\theta}}\left(a_t|s_t\right) + \gamma \sum_{a' \in A} \pi_{\bar{\theta}}\left(a'|s_{t+1}\right)\left(q_{\bar{\theta}}\left(s_{t+1}, a'\right) - \tau ln\pi_{\bar{\theta}}\left(a'|s_{t+1}\right)\right), \quad (1)$$

with $\pi_{\bar{\theta}} = sm\left(\frac{q_{\bar{\theta}}}{\tau}\right)$ is the softmax policy satisfying $\pi_{\bar{\theta}}\left(a|s\right) = \frac{exp\left(\frac{q_{\bar{\theta}}(s,a)}{\tau}\right)}{\sum_{a'} exp\left(\frac{q_{\bar{\theta}}(s,a')}{\tau}\right)}$, where τ is the temperature parameter scaling the entropy.

Hence, M-DQN is obtained by simply using $\hat{q}_{m-dqn}$ as the regression target of DQN.

B. What happens under the hood?

The authors of the paper [1] show two important properties of M-DQN that explain its improved performance:

1. It implicitly performs KL regularization between successive policies.

2. It increases the action-gap by a quantifiable amount. Let q_θ be the q-function of a given agent after training for some number of steps. We compute the empirical action-gap as the differences of estimated values between the best and second best action, $q_\theta\left(s_t, \hat{a}_t\right) - max_{a \in A \setminus \{\hat{a}_t\}} q_\theta\left(s_t, a\right)$, where $\hat{a}_t \in argmax q_\theta\left(s_t, a\right)$. The intuitive reason to want a large action–gap is that it can mitigate the undesirable effects of approximation and estimation errors made on q on the induced greedy policies.

III. Experiments (First Part)

We implement M-DQN as a variant of DQN from Stable-Baselines-3. For the hyperparamters used by both DQN and M-DQN, we use the same values for both agents. Since the log-policy term is not bounded, we make use of log-policy clipping; with a hyperparamter $l_0 < 0$ we replace $\tau ln\pi\left(a|s\right)$ by $\left[\tau ln\pi\left(a|s\right)\right]_{l_0}^{0}$. Hence, M-DQN has three additional hyperparamters compared to DQN: α, τ and l_0. We use the values reported in the paper to be working best, namely $\alpha = 0.9, \tau = 0.03$ and $l_0 = -1$.

We run our experiments using the implementation in the rl-baselines3-zoo repository from DLR. We train the DQN and M-DQN agents for 1 Million time steps on two Atari games: Breakout and Asteroids and show different plots of the two agents' behaviours, for comparison. Figures 1 and 2 show plots of training rewards (y-axis) w.r.t time steps (x-axis) with a moving window of 500 episodes of DQN and M-DQN on Breakout and Asteroids environments, respectively. In addition, Figure 3 shows the empirical action-gap of DQN and M-DQN computed during training on the Asteroids environment.

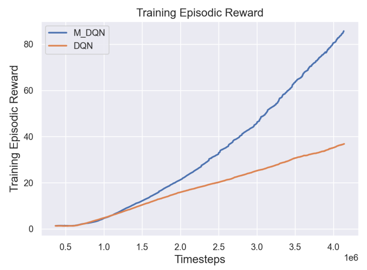

Fig. 1. Training reward for DQN and M-DQN on Breakout.

These plots show that M-DQN empirically outperforms DQN on Atari games. The natural question that arises now is whether this is also the case for robotic environments with continuous action spaces. This is the main concern of the second part of our project.

IV. M-RL for continuous action spaces

In the second part of the project, we wanted to extend the idea of Munchausen-RL to the continuous action space

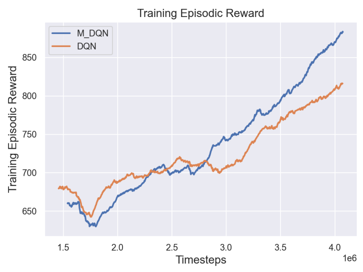

Fig. 2. Training reward for DQN and M-DQN on Asteroids.

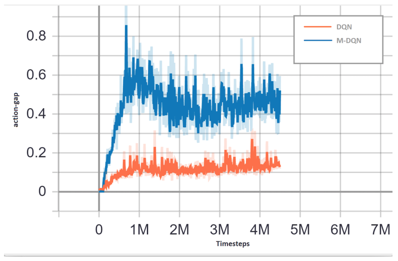

Fig. 3. Action-gap for DQN and M-DQN on Asteroids.

domain. So, we implemented the Munchausen-SAC algorithm as an extension to the popular SAC algorithm.

A. Key equations

The only change we bring to the original SAC implementation [3] is on the level of the Q-networks update equation. More precisely, the update target for Munchausen-SAC is given by:

$$y_{M-SAC}\left(r,s',d\right) = r + \left[\tau \alpha log\pi_\theta\left(a|s\right)\right]_{l_0}^{0} + \gamma\left(1-d\right) \\ \left(min_{j=1,2} Q_{\phi_{targ,j}}\left(s',\tilde{a}'\right) - \alpha log\pi_\theta\left(\tilde{a}'|s'\right)\right), \tilde{a}'\tilde{\ }\pi_\theta\left(.|s'\right), \tag{2}$$

where the red term is the only difference compared to SAC.

B. Continuous action-gap

Action-gap was originally defined for environments with discrete action space as the differnce between the best Q-value and the second best Q-value and described the confidence of an agent with the optimality of the choosen action. In the first part of the paper we have seen, that adding the Munchausen-term leads to increasing of the action-gap during the training, which means a decrease in the influence of approximation errors. Now we can not reuse this standard definition - even if we find the maximum value of approximated Q-function, the second best Q-value will be the same because of the continuity of the input space and neural network. Our definition of the action-gap for continuous action space is a mean of elementwise products of actor oriented and of critic oriented action-gaps.

1) Actor oriented action-gap: The goal is to measure the impact of the approximation error on the actor's decision. To achieve it we simulate the error by adding a random noise to all weights of the actor neral network, after that we generate the distorted action for given state and use the distance between real action, generated without noise, and the distorted action. This difference is the actor oriented action-gap.

2) Critic oriented action-gap: To measure the confidence of critic network we generate distorted action, as described above, and define the critic oriented action-gap as the difference between expected rewards for the optimal and distorted actions.

And now we define an extension of the action-gap in the given state for continuous action space:

Algorithm IV.1: STATEACTIONGAP($L, state$)

```
$ActionGap \leftarrow 0$
**for** $i \leftarrow 1$ **to** $L$
    **do** {
        $RealAction \leftarrow Actor(state)$
        $DistortedActor \leftarrow AddRandomNoise(Actor)$
        $DistortedAction \leftarrow DistortedActor(state)$
        $ActionGap \leftarrow ActionGap\ +$
            $Distance(RealAction, DistortedAction)*$
            $(Critic(state, RealAction) - Critic(state, DistortedAction)$
**return** $(ActionGap/L)$
```

Algorithm IV.2: ACTIONGAP(K, L)

```
$ActionGap \leftarrow 0$
$States \leftarrow SampleStates(ReplayBuffer, K)$
**for each** $state \in States$
    **do** $ActionGap \leftarrow ActionGap + StateActionGap(L, state)$
**return** $(ActionGap/K)$
```

High value of actor based action-gap means high uncertainty of actor about optimal action, high value of critic based action-gap means high confidence of critic about optimal action. Thus high value of the final action-gap shows that agent is more confident with his choice, which matches the original definition. As we can see on Figures 4 and 5, action-gap increases over time and correlates with reward of the agents.

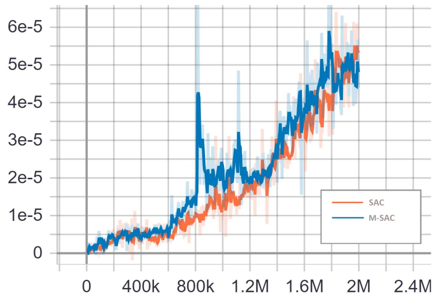

Fig. 4. Continuous action-gap (with parameters K = 2000, L = 200) for SAC and M-SAC on AntBulletEnv-v0.

V. EXPERIMENTS (SECOND PART)

A. Path planning task: Particles Environment

We decided to test the algorithms on a path planning task. We created a particles environment, where the agent wants to reach the goal, while avoiding the obstacles. Formally, the state space consists of the agent's velocity, the goal's position and the obstacles' positions, relative to the agent. The reward is always the negative distance of the agent to the goal plus the sparse signal of +10, when the agent reaches the goal, and -10 if it hits an obstacle. The action is the force to apply on the agent. This environment is illustrated in Figure 6.

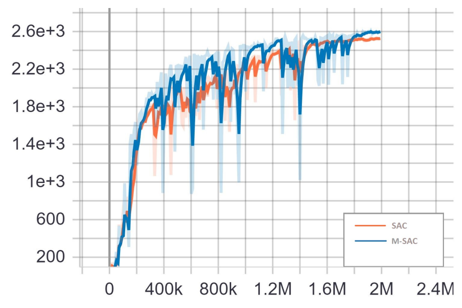

Fig. 5. Training reward for SAC and M-SAC on AntBulletEnv-v0.

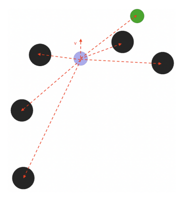

Fig. 6. Particles Environment

B. Results

We trained both SAC and M-SAC on this environment for 2M steps. For the hyperparameters of M-SAC, we set $\tau = 0.5$ and $l_0 = -2$. Then, we tested them on the same 10,000 configurations. When we tested them on random configurations, both agents performed very well, solving more than 98% configurations. But M-SAC performed slightly better than SAC. To confirm this observation, we took the same agents trained on the random configurations and tested them on somewhat harder configurations; instead of sampling the obstacles' positions uniformly at random, we sampled them from a Gaussian distribution, whose mean is sampled uniformly at random along the line connecting the agent and the goal. With this setting, the M-SAC outperformed SAC by more than 2%. The exact results of both agents are shown in Figure [7].

	random configurations	harder configurations (agents trained on random configurations)
SAC	98.48% (avg. time: **7.7**)	74.16% (avg. time: **11.9**)
Munchausen-SAC	**98.66%** (avg. time: 7.9)	**76.48%** (avg. time: 12.8)
Solved by both	97.52%	62.42%

Fig. 7. Results on Particles Environment

C. Particles Environment with fixed basis points

There are some limitations with the kind of state representation that we used so far. First, the agent can only operate in environments with a fixed number of obstacles. Second, there is no invariance to obstacles permutations. To fix these problems, we implemented an idea from the paper [4] that uses a fixed basis points set in a 3D vision task. In our setting, instead of giving the positions of the obstacles relative to the agent, we have some fixed number of basis points, which are unique in an agent's lifetime and for every one of these points, we give the position of the closest obstacle relative to this basis point as an observation to the agent. With this we can handle varying numbers of obstacles and also ensure invariance to obstacles permutation. On the other hand, the state space has higher dimensionality than the environment without basis points, if the number of obstacles is smaller than the number of basis points. The new environment is illustrated in Figure [8].

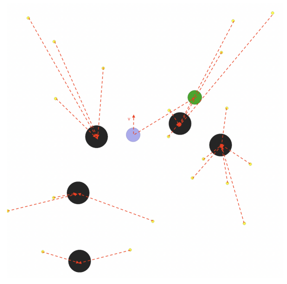

Fig. 8. Particles Environment with fixed basis points

D. Results

To test both agents on this new environment, we trained them on an environment with 5 obstacles and tested them on environments with 5 and 10 obstacles. Here, the gap is even larger between the two agents. And from the exact results in Figure [9], it is clear that M-SAC outperforms SAC.

	5 obstacles	10 obstacles (agents trained on 5 obstacles)
SAC	71.65% (avg. time: 7.3)	54,55% (avg. time: 6.71)
Munchausen-SAC	**77.14%** (avg. time: **7.2**)	**59,50%** (avg. time: **6.68**)
Solved by both	66.42%	48.03%

Fig. 9. Results on Particles Environment with fixed basis points

VI. Conclusion

Based on the experiments we have conducted in our project, it is clear that the simple idea of the Munchausen agent has several theoretical and practical benefits, when combined with different classical agents. We can confirm that the performance boost achieved by the Munchausen agent is present in environments with both discrete and continuous action spaces, with similar degrees.

References

[1] Vieillard, N., Pietquin, O., and Geist, M. (2020). Munchausen reinforcement learning. arXiv preprint arXiv:2007.14430.
[2] Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. A. 2013. Playing atari with deep reinforcement learning. CoRR abs/1312.5602..
[3] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In International Conference on Machine Learning.
[4] Prokudin, Sergey and Lassner, Christoph and Romero, Javier (2019). Efficient Learning on Point Clouds with Basis Point Sets. From Proceedings of the IEEE International Conference on Computer Vision, pages 4332-4341.