Data-Density guided Reinforcement Learning

Leon Lantz[1,†], Maximilian Schieder[1,†] and Michel Tokic[1,2,*]

1 - LMU Munich - Department of Computer Science
Geschwister-Scholl Platz 1, 80539 Munich - Germany

2 - Siemens AG - Foundational Technologies
Otto-Hahn-Ring 6, 81379 Munich - Germany

† - Contributed Equally

Abstract. This paper investigates reinforcement learning by avoiding low-density state regions using modified reward functions. The approach leverages data-density models within the state space, enabling a custom reward function that penalizes transitions into sparse regions. Applied in the Pendulum environment, this method encourages exploration in well-sampled areas while avoiding less-explored states. Empirical results show that this method effectively balances reward optimization with state confidence, enabling robust policy learning in challenging environments.

1 Introduction

Reinforcement learning (RL) is a widely used framework for optimizing the behavior of agents interacting with dynamic environments [7]. By iteratively interacting with their surroundings, agents learn an optimal policy based on feedback in the form of a numerical reward signal. A key distinction in RL lies between model-free and model-based approaches. Model-free RL relies on extensive real-world interactions to improve policies, while model-based RL uses predictive models of the environment to reduce the need for direct interactions. This makes model-based RL particularly beneficial in scenarios where real-world interaction is expensive or risky.

This paper addresses scenarios in which the agent must avoid states with low data density, i.e., states with sparse prior experience, leading to uncertainty in the predicted outcomes of the agent's actions. This is critical in applications such as heavy machinery, where operating in poorly understood configurations could result in significant risks or damage. Ensuring that the agent avoids such states, regardless of their associated rewards, is essential for safety and reliability. Typical offline RL approaches, such as Neural Fitted Q Iteration [5] or Deep Q-Learning [4], rely solely on samples for iteratively identifying the underlying Q-function. Through incorporating models instead, an agent can also explore states that were not present in the samples.

As investigated in this paper, the inclusion of a data-density model allows the agent to also account for the training dataset distribution when learning policies, with the goal of discouraging exploration in regions with low data density. This allows the agent to learn a policy that optimizes the original objective and avoids

[*]This contribution was supported with funds from the German Federal Ministry of Education and Research under project number 01IS24087A.

regions of high uncertainty. A well-established alternative would be integrating limits of authorities into policy training, which would require additional expert knowledge. Through experiments, we demonstrate that data-density based RL can balance reward optimization and state confidence, enabling robust policy learning in complex environments.

2 Methodology

Figure 1 illustrates the general architecture of our proposed methodology. In the following paragraphs, we provide a detailed description of each component in our model-based RL framework. One key component is the integration of the data-density model into the approach.

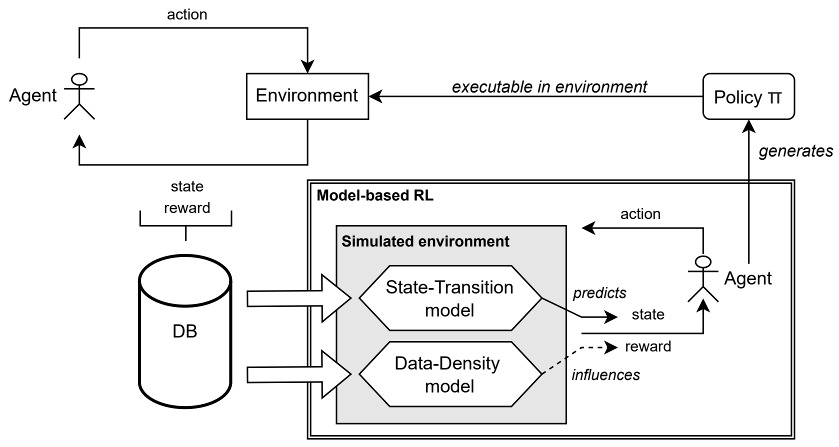

Fig. 1: Architecture of the proposed density-guided model-based RL approach.

We evaluate our approach using the Gymnasium Pendulum[1] environment [8]. Generally, a key requirement for environments suited to density-based policy learning is the presence of multiple viable paths to an optimal state. In the Pendulum environment, this requirement is exemplified by its ability to swing up in both a leftward and rightward direction.

Data Sampling - As illustrated in Figure 1, data plays a crucial role in enabling model-based RL, serving as the foundation for both the state-transition model and the data-density model. Such data is typically collected in the form of process data from industrial plants.

To evaluate the effects of sparse data on policy learning, an artificial low-density region is created in the state space of the Pendulum environment. Specif-

[1]`https://github.com/Farama-Foundation/Gymnasium/blob/main/gymnasium/envs/classic_control/pendulum.py` (Accessed: 2024-11-27)

ically, state samples within the angle range $[+2\pi/5, +3\pi/5]$ are excluded, representing an area where the agent's behavior remains uncertain. This range is chosen based on the pendulum's geometry to simulate conditions commonly arising due to limited exploration or environmental constraints. An episode is terminated once the agent enters a low-density region, ensuring that no experience is gathered in these areas. A total of 2000 episodes with a maximum of 200 steps per episode are simulated, using random actions to broadly cover the state space. This resulted in an overall dataset of 155.000 transitions. The collected data is split into training (70%) and validation (30%) sets.

State-Transition Model - The state-transition model represented in Figure 1 predicts the next state based on the current state and action, forming the core of any model-based RL framework. However, learning an accurate state-transition model is particularly challenging in scenarios where regions of the state space have sparse data coverage. In such cases, certain adaptations may be required, such as incorporating prior knowledge through physics-informed neural networks [6] or augmenting the training data with simulations. While these techniques can improve model accuracy, they also introduce additional complexities and uncertainties, particularly when simulated data deviates from real-world dynamics.

This paper focuses on using a data-driven approach for the state-transition model, trained exclusively on the available offline dataset. To predict state transitions, a Long Short-Term Memory (LSTM) [3] model is employed, leveraging its ability to capture temporal dependencies. Input sequences are created using a sliding-window approach, where four consecutive states and their actions serve as inputs, and the subsequent state is the prediction target. The LSTM architecture consists of 50 units followed by a dense layer. Training is performed using the Adam optimizer for up to 2500 epochs, with early stopping based on validation loss.

Data-Density Model - A central aspect of this work is leveraging data-density models to guide policy learning by distinguishing well-explored from less-explored regions of the state space, where behavior remains uncertain. These models approximate the underlying data distribution within the state space of the environment.

In our experiments, we evaluate Kernel Density Estimation (KDE) to model data density [1]. The KDE model utilizes the positional encoding of the pendulum's state. The bandwidth parameter, which controls the smoothness of the density estimate, is set to 0.1 based on cross-validation and visual control, balancing overfitting and underfitting. This density model is later integrated into the reward functions to penalize low-density states. While KDE is utilized here, the method is adaptable to alternative data-density estimation techniques.

Simulated Environment - A simulation of the actual environment is designed to facilitate model-based RL. In our experiments, this is based on a

Gymnasium environment, which incorporates the state-transition model into its `step()`-function. In contrast to the original Pendulum environment, which calculates states through physical equations using θ (angular position) and $\dot{\theta}$ (angular velocity), the custom environment predicts future states utilizing the state-transition model instead.

The environment supports multiple reward functions. The standard reward function serves as a baseline, while modified reward functions incorporate penalties for low-density states. Different penalty factors and thresholds are evaluated to assess their impact on policy training. Particularly, we modified the Pendulum's reward function with an additional penalty term p:

$$\text{reward} = \underbrace{-\left(\underbrace{\theta^2}_{\text{angle term}} + \underbrace{0.1 \cdot \dot{\theta}^2}_{\text{angular velocity term}} + \underbrace{0.001 \cdot \text{torque}^2}_{\text{torque term}}\right)}_{\text{original reward function}} - \underbrace{p}_{\text{penalty term}}$$

From the given reward function, the minimum possible reward within the bounds of the Pendulum environment can be derived. Considering the angle term $\theta^2 \leq \pi^2$, the angular velocity term $0.1 \cdot \dot{\theta}^2 \leq 0.1 \cdot 8^2$, and the torque term $0.001 \cdot \text{torque}^2 \leq 0.001 \cdot 2^2$, the overall minimal reward per step is approximately -16.27. This value is critical for balancing penalty and reward.

The `reset()`-function is also adapted to support flexible initialization of starting states. By default, starting states are randomly sampled from the unit circle, with angular velocities constrained to $[-2.0, 2.0]$.

Policy Training - Policy training aims to learn a control policy, π : `state` $\rightarrow$ `action`, that optimizes the agent's action selection strategy towards minimizing the reward penalty. Particularly, to stabilize the pendulum in its upright position.

We train policies using Soft Actor-Critic (SAC) from Stable Baselines because of its suitability for RL in continuous state and action spaces [2]. It should be noted that alternative algorithms, such as DDPG or PPO, could also be employed. However, they were not the primary focus in our study.

3 Evaluation and Results

The state-transition model was validated within the actual Pendulum environment. Comparisons between predicted and actual state trajectories confirm the model's capability to accurately approximate the pendulum's dynamics. As illustrated in Figure 2, the forecast error is higher in regions with no training data compared to well-sampled areas.

Subsequently, control policies are trained using SAC with default parameters in Stable Baselines. To evaluate the influence of penalizing low-density states, policies are trained with a density threshold of 0.025 and varying penalty values.

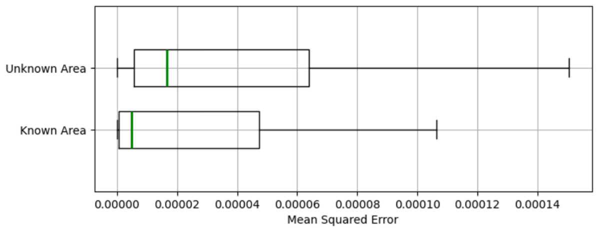

Fig. 2: Comparison of forecast errors (MSE) between regions with sufficient training data and those without.

This threshold was determined by analyzing different states in the data-density model. Additionally, the starting state during training is fixed at the resting position (pendulum hanging straight down, $\theta = \pi$, with $\dot{\theta} = 0$). Table 1 summarizes the evaluation results for different penalty levels after successful policy training.

p	**Left path (%)**	**Right path (%)**
2	41	59
10	36	64
30	15	85

Table 1: Results showing the proportion of left and right paths taken under varying penalty levels after 150.000 training steps. The numbers indicate how often the penalized area was visited through swing up within 2000 episodes.

We observe, as the penalty increases, the tendency to favor the right path becomes more visible (see Figure 3). By examining the original reward function at its boundaries, the underlying cause of this effect becomes evident, arising from the balance between the original reward components and the added penalty.

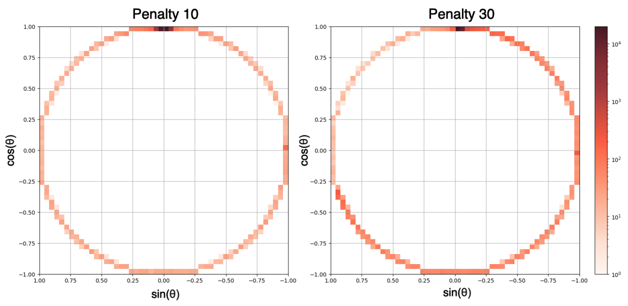

Fig. 3: Histogram of pendulum trajectories under penalty levels $p = 10$ (left) and $p = 30$ (right). We observe a more visible trend to favor the right path with increasing penalty p.

4 Conclusion

This work focused on integrating data-density models into reinforcement learning, aiming to improve policy learning by guiding agents to avoid regions with sparse data while optimizing their primary objectives. A state-transition model was trained to capture the environment's dynamics for model-based RL.

The study revealed that penalizing low-density states does not necessarily lead to their immediate and complete avoidance, given the maximum of 150.000 training steps in our experiments. Of course, the weighting of the penalty term needs to be selected appropriately according to the environment's reward function.

Further refinement of the reward function and additional computational resources could enhance these results. Furthermore, penalties could be applied dynamically based on density or by terminating episodes when density falls below a specified level. Balancing the trade-off between penalizing deviations from the optimal pendulum position and avoiding low-density regions is crucial for optimizing the environment's objectives while maintaining safety constraints.

The findings of this study provide valuable insights for applying density-aware reinforcement learning in real-world scenarios, especially in safety-critical applications. By building on these results, future work could explore the integration of alternative density estimation techniques and more sophisticated reward designs.

References

[1] Y.-C. Chen. A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology*, 1(1):161–187, 2017.

[2] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning (ICML 2018)*, pages 1861–1870, 2018.

[3] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation MIT-Press*, 1997.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[5] M. Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *16th European conference on machine learning (ECML 2005), Porto, Portugal, 2005.*, pages 317–328. Springer, 2005.

[6] M. A. Roehrl, T. A. Runkler, V. Brandtstetter, M. Tokic, and S. Obermayer. Modeling system dynamics with physics-informed neural networks based on lagrangian mechanics. *IFAC-PapersOnLine*, 53(2):9195–9200, 2020.

[7] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction.* MIT press, 2. edition, 2018.

[8] M. Towers, A. Kwiatkowski, J. K. Terry, J. U. Balis, G. D. Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, H. Tan, and O. G. Younis. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.