ReScience

[Re] Learning state representations with robotic priors

Loic Cressot[1], **Alexandre Coninx**[1], **Astrid Merckling**[1], **Nicolas Perrin**[1], **and Stéphane Doncieux**[1]

1 Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique, ISIR, F-75005 Paris, France

Lcressot@gmail.com

Editor
Name Surname

Reviewers
Name Surname
Name Surname

Received Month, day, year
Accepted Month, day, year
Published Month, day, year

Competing Interests:
The authors have declared that no competing interests exist.

Article repository

Code repository

A reference implementation of

→ Learning state representations with robotic priors, Jonschkowski, R. and Brock, O., 2015. Autonomous Robots, 39(3), pp.407-428

Introduction

Under Markovian assumptions, an agent interacting with an environment only needs its current observation in this environment to be able to optimally choose its next action. In this scenario, which we consider in this paper, state representation learning aims at learning to compress observations of an agent into compact and task oriented representations. By means of a dimensionality reduction and the removal of irrelevant information from observations, these compact representations should simplify the task of algorithms exploiting them, such as reinforcement learning algorithms. Jonschkowski and Brock [2] proposed a method for learning such a transformation by minimizing a loss with gradient descent. The loss implements priors that enforce the representations to be coherent with a physically plausible description of the robot state. The utility of the learned representations was evaluated using the fitted Q iteration reinforcement learning algorithm [1].

In this paper, we propose novel implementations of the state representation learning introduced in [2] and the fitted Q iteration algorithm, along with open-source code for the generation of new learning datasets [3][4]. We used these implementations to reproduce the main results of the original manuscript on a mobile robot navigation task in simulation. We did not reproduce the sections of [2] corresponding to a slot car racing task and a navigation task with a real robot.

Methods

We didn't contact the authors of [2], and as they did not provide code for the data generation and the fitted Q iteration algorithm, we reimplemented those two. This enabled us to generate our own training data and make our own testing of the trained policies using the learned representations as states for these policies. In addition, in order to maximize the independence of our results from those of the original paper, we also recoded the core algorithm with a different library: we used Keras with a Tensorflow backend instead of Sonnet and Tensforflow in the original code.

Data generation

Our simulator is an OpenAI gym environment [5] that can be easily installed with pip, the python package installer, and can be found on a public git repository [3] and a public archive [4]. Figures 1a and 6a display observations received in the simulator with an egocentric view, and Fig. 4a displays observations with an exocentric view (from above). In addition, Fig. 5 presents the environment with and without visual distractors.

Fitted Q iteration

We reimplemented the fitted Q iteration algorithm in Python, using exactly the same method and hyper-parameters as the one described in the original paper.

Representation learning

We implemented our own new version of the state representation algorithm in Python with the Keras library using a Tensorflow back end, and used the same hyper-parameters as in the original paper, except for the regularization parameter λ in the experiment with a view from above and distractors (see sections **Invariance to Perspective** and **Ignoring distractors**).

In three of the losses, conditional expectation are computed considering states, actions and rewards at two different time steps. In our opinion, the original paper has a minor ambiguity concerning the choice of the set of these time pairs t_1 and t_2. The original code shared by the authors of [2] allowed us to understand that for a given batch containing observations at different times t_i, all the possible pairs inside the batch are taken, provided that they satisfy some constraints present in the loss formulation (e.g. all the pairs (t_1, t_2) for which the actions are identical: $a_{t_1} = a_{t_2}$, or all the pairs with identical actions and different subsequent rewards: $a_{t_1} = a_{t_2} \wedge r_{t_1+1} \neq r_{t_2+1}$).

Results

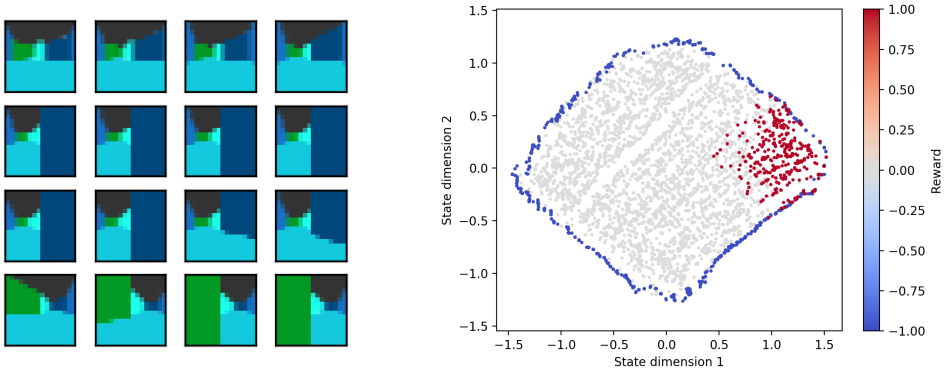
Simple navigation task

In this section we reproduce the first experiment presented in [2]: *the learning process on the simple navigation task*. As shown in figure 1a, we choose the parameters of our data generator that produce observations resembling to those chosen by the authors in this first experiment : one-colored walls with 300° field of view. Then we run our implementation of the state representation learning algorithm with the same hyper-parameters as in the article and found qualitatively similar learned 2D states, as shown in Fig. 1b.

We also run the same learning with 5 dimensional states and performed a principal component analysis (PCA) to show that the learned representation predominantly lies on only two principal dimensions (see Fig. 2), as observed in [2].

Afterwards, keeping the same training set, we run a 25 steps gradient descent training, and evaluate the learned representations at each step, as presented on figure 3. As in the article, we perform 10 fitted Q iteration trainings for each step of the representation learning, and for each of the policies learned with fitted Q iteration, we perform 20 evaluations in the same environment as the one used for generating the training data. We sum the rewards over each evaluation and average them for each fitted Q learning, and gather the points on the plot shown on figure 3. This experience shows that our results are not only qualitatively similar to those in the

paper but also quantitatively. We observe that the losses, stacked on figure 3, are less smooth than those in the article, particularly the causality loss. We also use a smaller gradient descent step than in the article because it seems that the representations are converging too fast for a proper step by step analysis.



(a) **(b)**

Fig. 1 *Simple navigation task experiment : (a)* Observations *(b)* 2D state representation learned.

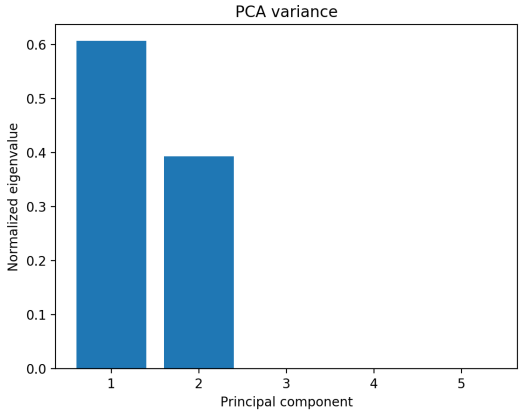


Fig. 2 *Eigenvalues on a 5D representation learned with the same data and parameters as in Fig. 1.*

Invariance to perspective

In this section we reproduce the second experiment of the article: *The invariance to perspective on the navigation task*. Again we use our own gym environment as a data generator to produce top-down perspective observations, as shown on figure 4a, where we artificially set a bigger radius to the robot, as in the article, so as to make it more visible. Then we run the state representation learning algorithm with the same hyper-parameters as the authors, except for the L_1 regularization parameter λ, see figure 4b.

Indeed, when setting the regularization parameter λ to a strong value such as $1e-1$ like in the paper, our implementation cannot learn any relevant representation, but converges instead to a blurry and mixed point cloud. However, for a low value

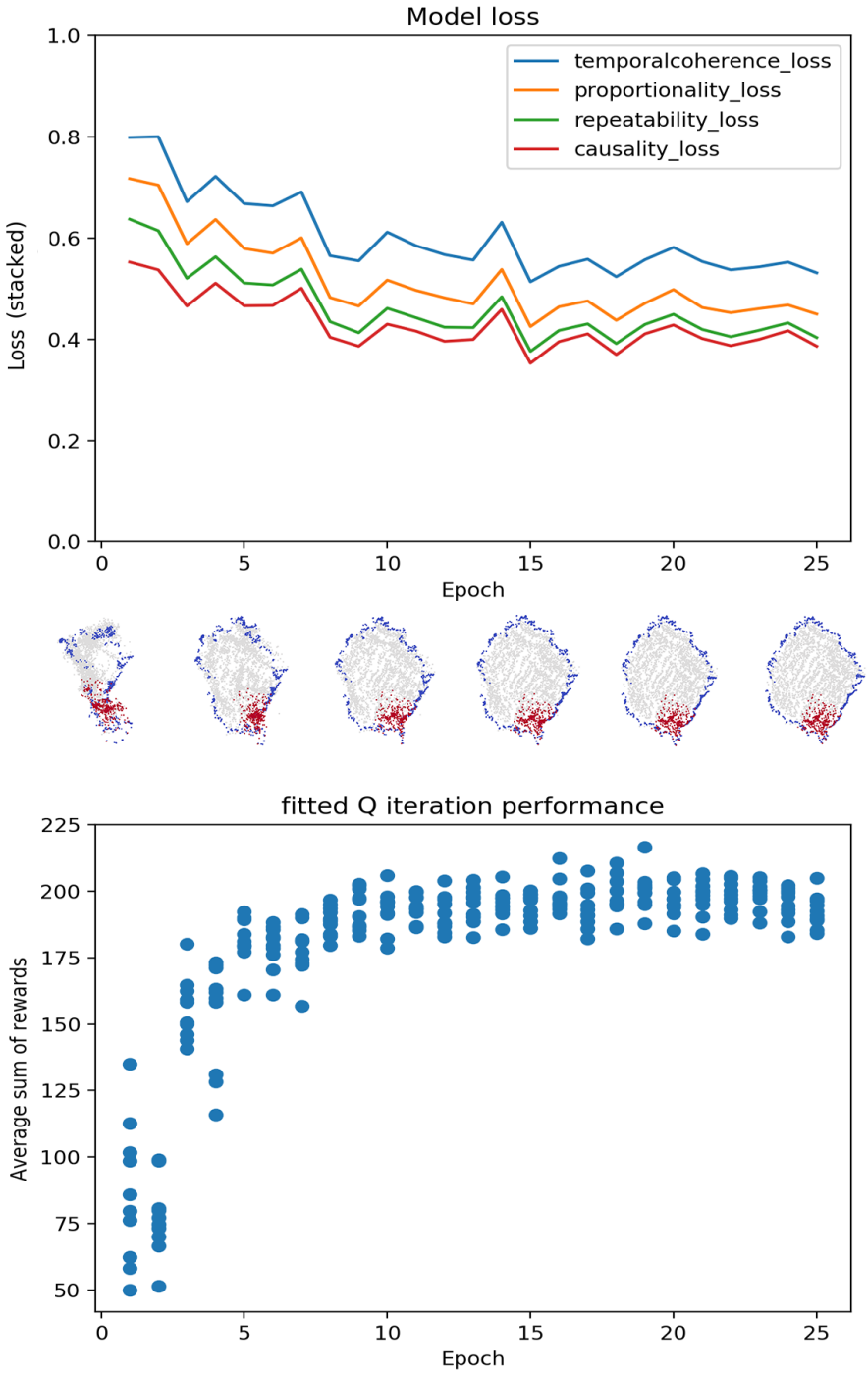
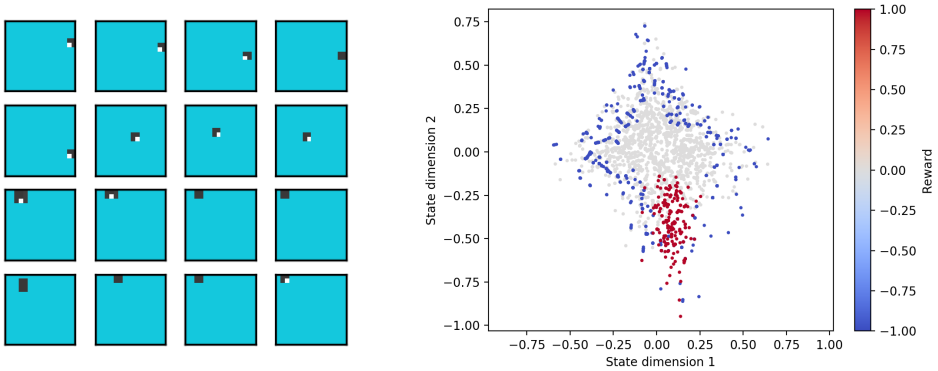


Fig. 3 *Fitted Q iteration evaluation*

of regularization, the learned representation is qualitatively similar to the one in the paper: a blurry version of the egocentric version. We should mention that we tried the initial code of the authors with the strong regularization value and it worked as presented in their paper.



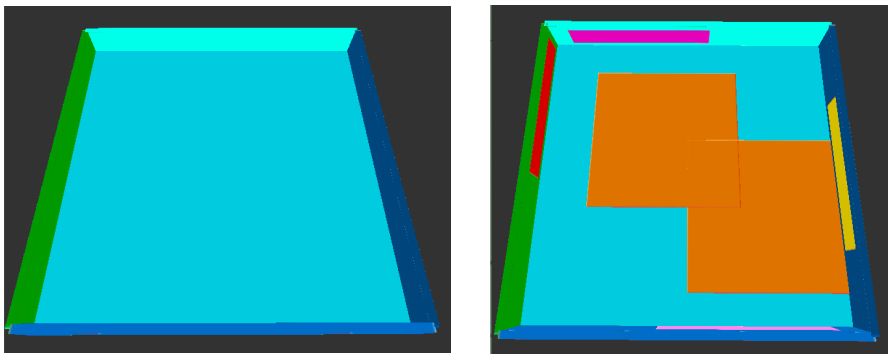
(a) (b)

Fig. 4 *Invariance to perspective experiment : (a)* Observations from a global point of view *(b)* 2D state representation with $\lambda = 1e-4$

Ignoring distractors

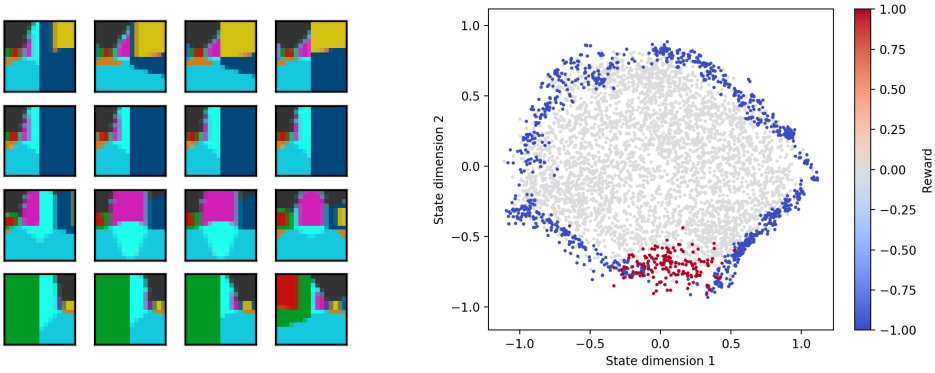
In this section we reproduce the experiment on the navigation task with visual distractors. As shown on figure 5, we implemented an option in our simulator to produce visual distractors with colored rectangles moving randomly on the surface of the walls and ground at a lower speed than the speed of the robot.

The obtained training observations, presented on figure 6a are used to train a state representation with the same hyper-parameters as in the paper, except again for the λ parameter which is set to the slightly lower value 0.1 instead of 0.3, simply because it seems to give a qualitatively more similar result. The learned representation, presented on figure 6b, is again coherent with the results presented in the paper.



(a) (b)

Fig. 5 *Adding visual distractors in the setup : (a)* Original setup *(b)* Setup with visual distractors



(a) (b)

Fig. 6 *Ignoring distractors experiment : (a)* Observations with visual distractors *(b)* 2D state representation with $\lambda = 0.1$

Conclusion

Our implementation produces results that seems faithful to those of the original paper. However, its computational speed is lower than the code provided by the authors, probably because we had to use some tricks to force the Keras library to deal with unsupervised learning and conditional expectations on pairwise operations.

References

[1] Damien Ernst, Pierre Geurts, and Louis Wehenkel. "Tree-based batch mode reinforcement learning". In: Journal of Machine Learning Research 6.Apr (2005), pp. 503–556.

[2] Rico Jonschkowski and Oliver Brock. "Learning state representations with robotic priors". In: Autonomous Robots 39.3 (2015), pp. 407–428.

[3] Cressot Loïc. gym-round_bot, an OpenAI gym environment of a turtle robot. 2018. URL: https://github.com/Lcressot/gym-round_bot.

[4] Cressot Loïc. gym-round_bot, an OpenAI gym environment of a turtle robot. 2018. DOI: 10.5281/zenodo.2538037. URL: https://doi.org/10.5281/zenodo.2538037.

[5] OpenAI gym environment interface. URL: https://gym.openai.com.