HAL open science

Density Independent Self-organized Support for Q-Value Function Interpolation in Reinforcement Learning

Antonin Calba, Alain Dutech, Jérémy Fix

To cite this version:

Antonin Calba, Alain Dutech, Jérémy Fix. Density Independent Self-organized Support for Q-Value Function Interpolation in Reinforcement Learning. 29th European Symposium on Artificial Neural Networks, Oct 2021, Bruges/Online, Belgium. 10.14428/esann/2021.es2021-62 . hal-03550442

HAL Id: hal-03550442

https://inria.hal.science/hal-03550442v1

Submitted on 1 Feb 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Density Independent Self-organized Support for Q-Value Function Interpolation in Reinforcement Learning

Antonin Calba[1] and Alain Dutech[2,3] and Jérémy Fix[2,4]

1- Université de Lorraine, Nancy - France.

2- Loria, Nancy - France ; 3- INRIA, Nancy - France
4- CentraleSupelec, Metz - France

Abstract.
In this paper, we propose a contribution in the field of Reinforcement Learning (RL) with continuous state space. Our work is along the line of previous works involving a vector quantization algorithm for learning the state space representation on top of which a function approximation takes place. In particular, our contribution compares the performances of the Kohonen SOM and the Rougier DSOM with the Göppert function approximation scheme on both the mountain car problem. We give a particular focus to DSOM as it is less sensitive to the density of inputs and opens interesting perspectives in RL.

1 Introduction

Reinforcement Learning (RL) methods have gained in popularity in the last few years, mainly thanks to end-to-end deep learning approaches, especially in the context of value function approximation [1]. In the deep learning approaches, the state directly feeds in the neural network approximating the function and the representation is learned and driven by the TD error backpropagated through the network.

This paper contributes to another approach to approximation by exploring the use of interpolation methods where the set of support points of the interpolation is self-organized using vector quantization algorithms, such as Self-Organizing Maps (SOM) [2] and Dynamic Self-Organizing Maps (DSOM) [3]. Using a topological representation of the state space as in SOM and DSOM allows to consider function interpolation algorithms which exploit that topology, as in the CI-SOM algorithm of Göppert [4] for example. Another advantage lies in the possible hindsight gained from analyzing the position of the reference vectors of the quantization, for example revealing portion of the state space that are frequently visited or giving us more information on the structure of the problem.

The SOM algorithm is, as several other quantization algorithms, strongly dependent on the density of the inputs feeding it. In the context of (RL), this means that the dynamics of the environment drives the density of the reference vectors. However, it might be that the dynamics of the environment makes the agent wanders around in some part of the state space which is not necessarily very interesting, informative and any case does not necessarily deserve to be

over represented by a lot of reference vectors. Indeed, the regions of the space that the agent encounters more frequently are not necessarily the regions where it is the most difficult to learn the value function. DSOM, an alternative of SOM, which is less sensitive to the density of the inputs has been proposed in [3] and relies on a slightly different update functions of the reference vectors. This opens a way to more flexibly control where it is relevant to allocate the limited resources of the reference vectors.

The major contributions of our paper are 1) using DSOM as a vector quantization method in the context of RL using Göppert's interpolation, 2) comparing SOM and DSOM merits in that context. To that end, we first present some background materials and our methodology (sec 2), then give some results on the classical Mountain-Car benchmark (sec 3) and conclude with some discussion in the light of related works (sec 4).

2 Materials and methods

In the Reinforcement Learning paradigm [5], an agent confronted to a situation s_t from state space $\mathcal{S}$ at time t must learn to choose the "best" action a_t to optimize its long term discounted reward sum ($\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$) where scalar reward r_t is received after each state transition ($\gamma \in [0, 1)$ being a scalar discount). Q-Learning, a classical algorithm, learns an optimal policy $\pi^*(s_t) = \text{argmax}_{a \in \mathcal{A}} Q^*(s_t, a)$ where the action-value function $Q^*()$ is updated after each (s_t, a_t, r_t, s_{t+1}) transition using

$$Q^*(s_t, a_t) \leftarrow Q^*(s_t, a_t) + \alpha \left[r_t + \gamma \max_a Q^*(s_{t+1}, a) - Q^*(s_t, a_t) \right]. \quad (1)$$

When the state space is large or continuous, approximation methods must be used to represent the action-value function $Q^*()$.

In this paper, we work with continuous state space $\mathcal{S}$ and discrete action spaces $\mathcal{A}$. Therefore, the value function can be learned by approximating the set of value functions $\{Q^*(., a), a \in \mathcal{A}\}$. We perform approximations by using vector quantization to learn a representation (a set of support vectors) of the state space $\mathcal{S}$ from which we run interpolation algorithms described later. While the agent is operating in the environment, its states $s_t \in S$ feed a vector quantization algorithm, SOM or DSOM in our experiments.

Neurons in SOM and DSOM are topologically related (in a grid lattice for example) with each other, each one embeds a prototype (or reference vector) lying in space $\mathcal{S}$. In all our experiments, inputs are normalized in $[0, 1]$. The Self-Organizing Map algorithm[2] proceeds by finding the best matching unit[1] (BMU) indexed i^* among the reference vectors w_i and updates every reference vector depending on its distance in the SOM lattice to the BMU according to :

$$\forall i, w_i \leftarrow w_i + \epsilon_S h_\sigma(i, i^*)(s_t - w_i) \quad (2)$$

[1]The BMU is the unit which has its reference vector closest to the current input

with the neighborhood function defined as $h_\sigma(i,j) = \exp(-\frac{d(i,j)^2}{2\sigma_h^2})$ where $d(i,j)$ is the distance in the SOM lattice between units i and j. In our experiments, ϵ_D (resp. σ) decreases geometrically from 0.5 to 0.01 (resp. 1.0 to 0.1).

We also consider DSOM [3] which has a slightly different update function :

$$\forall i, w_i \leftarrow w_i + \epsilon_D \|s_t - w_i\|_{\mathcal{S}} h_\eta(i, i^*, s_t)(s_t - w_i) \tag{3}$$

with the neighborhood function defined as $h_\eta(i, i^*, s) = \exp(-\frac{d(i,i^*)^2}{\eta^2 \|s_t - w_{i^*}\|_{\mathcal{S}}})$. In other words, there is not much learning when the reference vector of the BMU w_{i^*} is sufficiently close to the current input s_t. Therefore the $\frac{1}{\|s - w_{i^*}\|_{\mathcal{S}}}$ term prevents the reference vectors from aggregating in the case of densely packed inputs. In our experiments, ϵ_D is fixed at 0.1 and η at 10.

In the context of reinforcement learning, we use either SOM or DSOM to quantize the state space $\mathcal{S}$ and add a Q-table with one entry $qt_i(a)$ for each action to every neuron i. A basic piece-wise constant interpolation function is thus $\hat{Q}_p(s,a) = qt_{BMU(s)}(a)$ and the Q-table of the BMU is updated after each transition using eq (1), with α decreasing from 0.1 to 0.001. A more complex interpolation function $\hat{Q}_g$ has also been tested using Göppert's interpolation [4] in an evaluation phase.

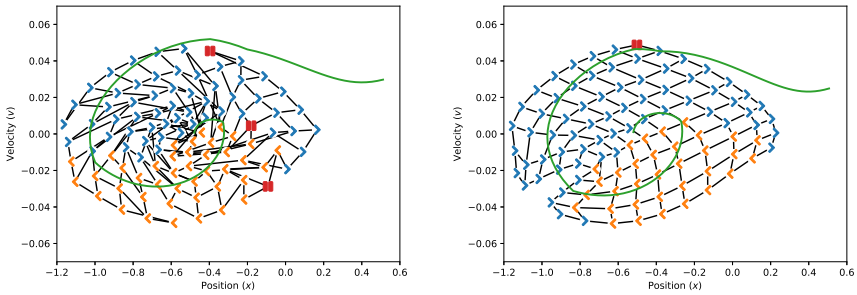

Fig. 1: Position of and best action ('<' for `left`, '>' for `right`, '||' for `none`) of the support vectors at the end of learning. Left plot: SOM. Right plot: DSOM. For both, we plot in green the trajectory of a greedy policy starting from the bottom of the valley.

The interpolation algorithm of [4] works in two stage as an interpolated I-SOM is smoothed with Shepard's functions. I-SOM value for a state s is a weighted average of neighboring units QValues through the use of local coordinates in the map lattice. This interpolation leads to local linear interpolation with discontinuities that is smoothed with inverse distance weighting functions (Shepard's function) in CI-SOM which we call the Göppert scheme.

3 Results

3.1 Mountain car

We run experiments on the Mountain Car of the OpenAI Gym set of benchmark environments [6]. SOM and DSOM maps are arbitrarily organized in 10×10 grids. Each experiment is run for 1.000 episodes with a fixed max number of iterations. During the first 1/3 episodes, the learning agent follows a uniform random policy and during the last 2/3 episodes, a random Boltzmann policy based on the current piece-wise constant QValue function $\hat{Q}_p(s, a)$. At fixed intervals and at the end of learning, a deterministic greedy policy is derived from the SOM or DSOM mapping in order to compare performances.

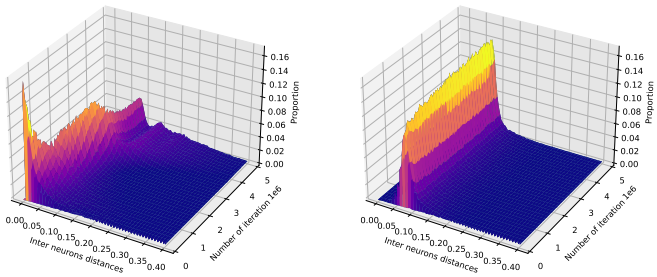

Fig. 2: Histograms of the normalized distances between the neighboring neurons in the lattices for SOM (left) and DSOM (right).

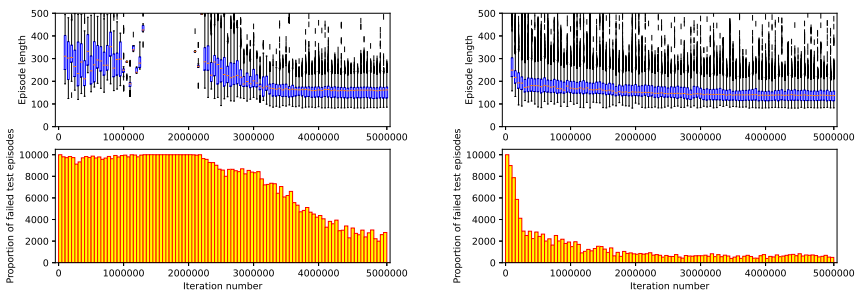

Fig. 3: Statistics of the evaluation episodes for both SOM (left) and DSOM (right). The top plot is a boxplot (median, first and third quartiles) of the length of episodes lasting less than 500 iterations. The bottom plot is the number of failed evaluation episodes. Results are over 100 tests episode every 100 training episodes.

Experiments with SOM and DSOM both converge to a near optimal policy that reach the goal in approximately 130 steps. But, as shown on Figure 1, the neurons (and thus the support vectors) are better deployed when using DSOM. In addition, as shown on figure 3 where average performance and the

average number of failed trajectories are presented, it seems that DSOM allow faster learning than SOM. This is mostly due to DSOM better unsupervised deployment of the support vectors, as illustrated on Figure 2 where inter-neurons distance are more varied for SOM. Note also that, in the case of SOM, two peaks on distances appear at the end corresponding to a twist of the SOM map and apparent on figure 1, left.

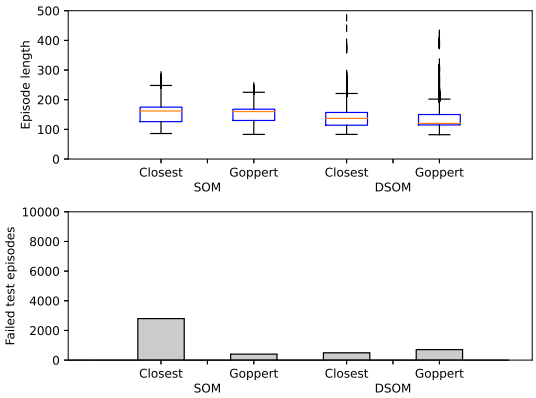

Fig. 4: At end of training, comparison of both interpolation function. Piece-wise constant (`Closest`) using $\hat{Q}_p$ and Göppert (`Goppert`) using $\hat{Q}_g$.

Interpolation using Göppert computation scheme is costly and we use it on the support vectors obtained at the end of learning to get an interpolated QValue function $\hat{Q}_g(s, a)$ and compare greedy policies derived either from that interpolated function or the basic piece-wise function $\hat{Q}_p(s, a)$. Figure 4 shows that Göppert interpolation could be really useful for the SOM, maybe compensating for the difficulties of SOM to map the state space.

4 Discussion

Our experiments on the Mountain Car benchmark confirmed other works ([7, 8, 9]) demonstrating the use of self-organized maps, SOM or DSOM, to learn an adaptive state representation in RL. To our knowledge, it is the first time that this Dynamic-SOM (DSOM) has been explored in this setting even though its main characteristic - a lower sensitivity to the density of training samples - seems to be a strong asset in reinforcement learning. Indeed, is it very frequent in RL that the "decisive" states are rarely visited compared to more trivial less important states. Classical SOM is likely to focus on a good representation of these frequently visited states detrimental to rarer and more important states. It is particularly the case with the Mountain Car where states at the bottom of the valley are over visited as long as the agent has not learned to reach the goal. Also, SOM had more trouble unfolding over the state space than DSOM.

Still, our work can be considered as preliminary. Exploring the influence of the hyper-parameters, in order to finely tune them, is still to be done. We should also explore the influence of the underlying lattice structure. We should also consider other environments, with other characteristics, to really assess the strength and weaknesses of DSOM. For example, our current experiments on the acrobot problem tend to give a slight advantage of SOM over DSOM.

In the future, we would like to explore more deeply how to intertwine interpolation of the QValue and self-organizing maps. If more complex interpolation schemes like Göppert may compensate for the situation where classical SOM do not unfold easily, we would like to test its usage during learning, to see if it can accelerate learning. Beside, instead of using a 2D regular grid of fixed size, it could be interesting to use an adaptive structure like in *growing SOM* (see [10, 11]) to better fit the reachable state space. As in [12], DSOM could also be used to map the QValue function space instead of the state space. But, in the short term, our most likely focus will be on using the current QValue approximation to try to guide learning, either by orienting the agent to states where the QValue still has a big variance or even by modulating the self-adaptation rules with some criteria derived from the QValue.

References

[1] David Silver et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

[2] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982.

[3] Nicolas Rougier and Yann Boniface. Dynamic self-organizing map. *Neurocomputing*, 74(11):1840–1847, 2011.

[4] Josef Göppert and W Rosentiel. The continuous interpolating self-organizing map. *Neural Processing Letters*, 5(3):185–192, 1997.

[5] R. Sutton and A. Barto. *Reinforcement Learning*. Bradford Book, MIT Press, Cambridge, MA, 1998.

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[7] Claude F. Touzet. Neural reinforcement learning for behaviour synthesis. *Robotics and Autonomous Systems*, 22(3):251–281, December 1997.

[8] Andrew James Smith. Applications of the self-organising map to reinforcement learning. *Neural Networks*, 15(8):1107–1124, October 2002.

[9] Hesam Montazeri, Sajjad Moradi, and Reza Safabakhsh. Continuous state/action reinforcement learning: A growing self-organizing map approach. *Neurocomputing*, 74(7):1069–1082, 2011.

[10] Michael Baumann and Hans Kleine Büning. Adaptive function approximation in reinforcement learning with an interpolating growing neural gas. In *2012 12th International Conference on Hybrid Intelligent Systems (HIS)*, pages 512–517, 2012.

[11] Maxime Guériau, Nicolás Cardozo, and Ivana Dusparic. Constructivist approach to state space adaptation in reinforcement learning. In *2019 IEEE 13th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pages 52–61, 2019.

[12] Thommen George Karimpanal and Roland Bouffanais. Self-organizing maps for storage and transfer of knowledge in reinforcement learning. *Adaptive Behavior*, 27(2):111–126, April 2019.