[Re] A neurally plausible model learns successor representations in partially observable environments. NeurIPS Reproducibility Challenge 2019

Pranav Mahajan*
Department of Electronics and Communications
BITS Pilani, Goa Campus.
f20170277@goa.bits-pilani.ac.in

1 Introduction

Successor representations (SR) have been proposed as a middle-ground between model-based and model-free reinforcement learning strategies. Model-based learning allows for goal directed behaviours and are flexible to the changes in environment or reward functions whereas model-free learning allows for rapid actions. It's been suggested that the brain makes use of both of these approaches and features from neural responses are consistent with the SR framework. SRs allow for fast value computation as well fast adaptation to reward or goal location changes. The original paper shows how SRs can be computed in continuous, noisy and partially observable environments. The approach they present utilises Distributed distributional codes (DDC) which are a candidate for neural representation of uncertainty. The framework for learning distributional successor features which is presented by the original paper is consistent and relates well with hippocampal literature. For example, it has been argued that hippocampus holds the internal model of the environment, this framework addresses the neurally plausible ways in which such a model can be acquired. For more information kindly we refer to the original paper[1].

In our work, we reproduce the baseline results from the original paper partially and address possible reasons to as to why we couldn't reproduce some results. Being a theoretically intensive paper, we also emphasize on resources that were helful in familiarizing with background in DDC and SRs. Since the code wasn't open sourced or publicly available at the start of the NeurIPS reproducibility challenge and eventually authors open sourced a repository with core functions used in their work, we also address possible hurdles in exact reproduction of the results in the original paper from scratch i.e. attempting to reproduce the results by just reading the paper and supplementary materials.

2 Background

2.1 Partially observable Markov decision processes (POMDP)

In case of POMDPs, the transitions between states is probabilistic, we can control transitions using actions but the states are not completely observable (in comparison to MDP). Thus in a POMDP setting the agent needs to compute posterior beliefs μ as the agent doesn't have access to the states s. In a sense, it needs to infer this from the observations. Many a times, these beliefs μ_t are updated recursively as they depend on belief at a previous timestep μ_{t-1} and observation at the current timestep o_t.

*The code is available at `https://github.com/PranavMahajan25/NeurIPS-repros`

Preprint. Under review.

2.2 DDC wake-sleep

In a Distributed distributional code (DDC), a population of neurons represent distributions in their firing rates implicitly, as a set of expectations $\mu = E_{p(s)} = [\psi(s)]$ where μ is vector of firing rates, $p(s)$ is the distribution and $\psi(s)$ is a vector of encoding functions specific to each neuron. For all practical purposes, we believe it's fine to refer to DDC as distributions represented by set of expectations. The authors use an wake-sleep algorithm similar to that used to train a Helmholtz machine. The wake-sleep algorithm will be discussed in detail in section 3. We noticed quite less papers linking the ideas of DDC, wake-sleep and learning the internal model in our brain, we found this tutorial on Helmholtz machine [2] very helpful along with the previous work by the authors on DDC Helmholtz machine. In the original paper, DDC is used by the authors in the process of filtering the posterior when recognition model makes an estimate $q(s_t|o_t)$ of all previous posterior distributions $p(s_t|o_t)$. Thus the beliefs from the observations can be written as $\mu_t(O_t) = E_{q(s_t|O_t)}[\psi(s_t)]$ where $O_t : (o_1, o_2, ...o_t)$.

2.3 Successor representations from features

SR can also referred to as Successor feature representation or SF interchangeably. Authors describe a method to train these distributional SFs and use them in value computation an decision making. In a classical sense, SR for a state s_i is defined as the expected discounted sum of future occupancies for each state s_j, given the current state s_i. Thus the SR matrix $M(s, s')$ is a matrix associating each state to every other state. In case of continuous environments, we need to deal with curse of dimensionality, we express states in terms of set of K features $\psi_i(s)|_{i=1}^K$, thus SF for a state only needs to predict the discounted probability that a feature ψ_j will be observed in the future and not the complete state representation. Authors follow linear function approximation throughout their study and with same set of features. Thus most expressions are analogous to the discrete states form. The SF matrix M and the value function can be written as follows,

$$M^\pi(s_t, i) = E[\sum_{k=0}^{\infty} \gamma^k \psi_i(s_{t+k})|s_t, \pi] \approx \sum_j U_{ij}\psi_j(s_t) \tag{1}$$

$$V^\pi(s_t, i) = w_{rew}^T M^\pi(s_t) \tag{2}$$

U_{ij} can be found by temporal difference learning. It's important to note that the TD error is the prediction error of state features and not rewards.

2.4 Neural substrates of successor representation

Just to emphasize on how the work by authors is neurally plausible and connects with what we know from neuroscience is that authors in [3] have shown that feature-specific prediction errors are detected in fronto-striatal network and thus dopamine can also hypothesized to play a role in the state prediction error. It's also hypothesized that hippocampus might be where the SR matrix may be present [4, 5] and thus directs towards thinking of hippocampus as a predictive map. Thus this study addresses the learning in the wide spectrum between model-based and model-free methods.

3 Learning Process

In this section, we'll go over the wake-sleep algorithm in much more detail and include minor details which are ablated from the original paper but are necessary for reproduction of the results. Wake-sleep algorithms are unsupervised learning algorithms often used as probability density estimators, the goal of the wake phase is to learn the generative model whereas the goal of the sleep phase is to learn the recognition model.

The role of the wake phase is in two parts, first being finding $p(s_{t+1}|s_t)$ which can be boiled down to learning the transition matrix T. Second being, inferring posterior beliefs μ from all previous observations $O_t : (o_1, o_2, ...o_t)$. This is an estimate of the posterior distribution and is done by running the filter that whose weights are being learnt in the sleep phase.

The role of the sleep phase is to learn a recursive function f_W for updating beliefs μ. This can also called as learning the filtering weights. Having explained the role of wake and sleep phase, we'll head into the algorithmic implementation details.

Algorithm 1: Wake-sleep model in DDC state-space model

Result: Learnt weights T and W
Learn α_s weights to decode s from $\psi(s)$
Initialise T and W (random)
while *not converged* **do**
  Sleep phase:
  Generate sleep samples: $\{s_t^{sleep}, o_t^{sleep}\}_{t=0...N} \sim p(S_n, O_n)$
  Encode as a set of features: $\psi(s_t^{sleep}), \psi(o_t^{sleep})$
  Update filter weights W: $\Delta W \propto \sum_t (\psi(s_t^{sleep}) - f_W(\mu_{t-1}(O_{t-1}^{sleep}, o_t^{sleep}))\nabla_W f_W$
  Wake phase:
  $O_n \leftarrow$ collect wake observations
  Encode as set of features: $\psi(o_t)$
  Run the filter to infer posterior: $\mu(O_t) = f_W(\mu_{t-1}(O_{t-1}, o_t))$
  Update T: $\Delta T \propto (\mu_{t+1}(O_{t+1}) - T\mu_t(O_t))\mu_t(O_t)^T$
  Update observational model parameters
end

DDC takes distributional features as input which need to be extracted from latent or observed states rolled out from random walk. After referring to the supplementary materials and the open sourced code, it was clear that for latent and observed states, $K = 100$ Gaussian radial basis function features with their centres arranged on a 10x10 grid are used. The features are truncated along the internal walls.

A minor hurdle was generating samples in sleep phase. For implementation purposes can be interpreted and broken down into smaller steps as follows,

$$p(s_{t+1}^{sleep}|s_t^{sleep}) = \alpha_s T\psi(s_t^{sleep}) + \sigma_s * \eta/||\eta|| \tag{3}$$

where α_s are the regression weights to decode state s_t from distributional features $\psi(s_t)$. This is an ablation from the paper which is crucial for reproduction of the results. An alternative modification to do linear readout from DDC can be as follows and the weights will be learnt accordingly,

$$p(s_{t+1}^{sleep}|s_t^{sleep}) = \alpha_s T(\psi(s_t^{sleep}) + \sigma_s * \eta/||\eta||) \tag{4}$$

The corresponding sleep observation as,

$$p(o_t^{sleep}|s_t^{sleep}) = s_t^{sleep} + \epsilon \tag{5}$$

where ϵ, η, σ_s same as in section 4.1.

It's important to note that we need to have a decoding strategy $s \approx \alpha_s\psi(s)$ before generating sleep samples. We train a ridge regression on 30,000 sample observation trajectories to get α_s prior to training the wake-sleep algorithm. These are a few minor clarifications regarding ablations from the paper. While verifying the approximations in the wake-phase T update from the supplementary materials, it's important note that we are approximating KL divergence by Euclidean distance, keeping in mind KL is not a symmetric function and we are approximating it with a symmetric euclidean distance.

Since we are assuming approximately linear relationship throughout, the weights of the filter are trained using a ridge regression with a suitable $\tau = 1e - 4$. Once the wake-sleep algorithm is trained for 50 epochs we can draw inference reproduce results from the Fig. 1 of the original paper. We can learn distributional SFs in the sleep or the wake phase or else compute them from the dynamics after the wake-sleep algorithm has converged. SF matrix can be learnt in a variety of ways and we compare those in the next section.

In case of value computation, reward vector w_{rew} and the SF matrix M^π under a policy needs to be independently calculated and we can then compute values V^π by equation (2). We couldn't re-implement value computation under noisy environments due to time constraint.

Figure 1: Observed trajectories and true latents.

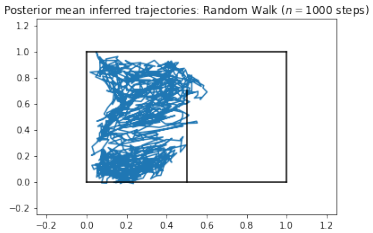

Figure 2: Posterior mean inferred trajectories.

4 Implementation and results

4.1 Environment implementation

A continuous [0,1] x [0,1] environment is implemented with vertical or horizontal internal walls in between. Random walk policy is implemented for the generative model. The agent bounces off (reflects) a wall if the random walk roll out crosses a wall. Random walk can be implemented as sampling the steps in X and Y direction from a Gaussian distribution as per the following, where $\eta \sim \mathcal{N}(0, \sigma = 1)$ and $\sigma_s = 0.06$

$$p(s_{t+1}|s_t) = [s_t + \sigma_s * \eta/||\eta||]_{WALLS} \tag{6}$$

Alternatively, random angle walk can also be implemented where the angle for each step is sampled from $[0, 2\pi]$ and a step of constant magnitude σ_s is taken in that direction. The observations are generated by adding a Gaussian noise (σ_o).This is necessary as in a POMDP, agent receives observations that depend on current latent state via an observation process. Thus the paper formally writes it as follows, where $\epsilon \sim \mathcal{N}(0, \sigma_o = 0.1)$.

$$p(o_t|s_t) = s_t + \epsilon \tag{7}$$

The RBF features with centres arranged on a grid have width $\sigma_\psi = 0.3$

We felt a lack of widely used environments like OpenAI Gym for a scenario similar to the one presented in the paper. Development of such new environments will certainly help lower the barrier for this field. Training can be done on a single CPU setup and in less than half and hour.

We were able to re-implement the wake-sleep algorithm and reproduce the results from Fig. 1 of the original paper. We re-implemented the functions for learning distributional successor features but we weren't able to reproduce the results of the value computation under random walk policy and discuss possible reasons for the same.

4.2 Learning and inferencing the state-space model parametrised by DDC

We train a state-space model corresponding to a random walk policy in the latent space with noisy observations using the DDC algorithm. We learn the decoding from $\psi(s)$ to s prior to wake-sleep training as explained in section 3. Prior inferring trajectories, we trained a decoding from μ features to true latent trajectories. The decoding scheme will directly affect reproduction of the results but is ablated from the paper. We request the reviewers to verify this approach as using the same $_s$ which were used to decode $\psi(s)$, now if used to decode from μ didn't give right results.

It was slightly unclear as to how to replicate the visualisation in Fig. 1(b) from the original paper. The supplementary material briefly describes it as approximating mean dynamics as a linear readout from DDC: $E_{s_{t+1}|s_t} \approx \alpha T \psi(s_t)$ where $s \approx \alpha \psi(s)$.

We implement and reproduce the results from Fig. 1 (c) of the original paper and are shown in Figure. 1 and Figure. 2

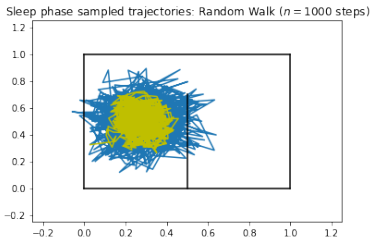

Figure 3: Sleep phase sampled trajectories

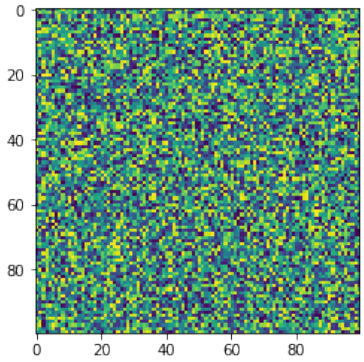

Figure 4: SF matrix visualization.

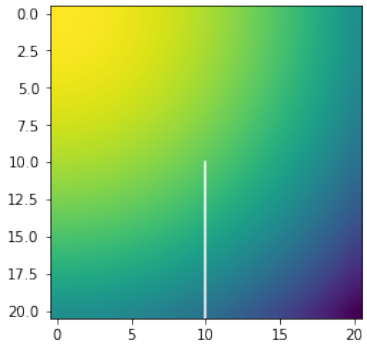

Figure 5: Reward function

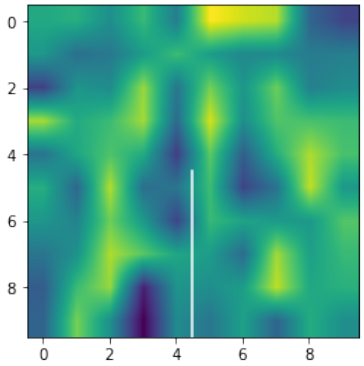

Figure 6: Value function from true latent

4.3 Learning Distributional SF

As shown in the original paper, we can compute successor features in closed form in the latent space and can be formally written as follows,

$$E_{s_t|O_t}[M(s_t)] = (I - \gamma T)^{-1} E_{s_t|O_t}[\psi(s_t)] = (I - \gamma T)^{-1} \mu_t(O_t) \tag{8}$$

The learnt 100x100 SF matrix can be visualised as shown in Figure. 4 but it doesn't help much.

4.4 Value functions using SFs under a random policy

SF matrix was learnt in sleep phase We defined the reward function as $r = 2 - (2x)^2 - (2y)^2$ and then sampled 100 latents from sleep phase. We then get the reward vector $w_{rew} = (r(\psi(s^{sleep})^{-1})^T$. We use this to generate observed reward weights from sleep observations. We plot evaluate value function using equation (2) at the same grid used for 100 RBF features and interpolate to get the Figure 6. We request the reviewers to verify this approach as the process to get those plots were ablated from the paper.

5 Conclusion

In this work we have presented a reproducibility analysis of the NeurIPS 2019 paper "A neurally plausible model learns successor representations in partially observable environments" by Eszter Vertes and Maneesh Sahani. We first attempted by relating to discrete completely observable environments, but communication with authors made sure that DDC SFs are not meant to propose an improvement in that scenario but address an interesting problem of learning in noisy continuous partially observable environments. By making reasonable assumptions we could only partially

reproduce the results but couldn't reproduce value computation, we believe due to mistake on our side in the method calculation of reward vectors. Also we believe the authors can explain the decoding scheme to avoid confusion and if possible release a well documented code so that it'll be easier to build upon their work. We couldn't implement value computation under noisy environment due to time constraints.

Acknowledgments

We would like to thank the author Eszter Vertes for her helpful clarifications and thank CodeOcean for compute.

References

[1] Eszter Vertes and Maneesh Sahani. A neurally plausible model learns successor representations in partially observable environments. *arXiv preprint arXiv:1906.09480*, 2019.

[2] Kevin G Kirby. A tutorial on helmholtz machines. *Department of Computer Science, Northern Kentucky University*, 2006.

[3] Mariann Oemisch, Stephanie Westendorff, Marzyeh Azimi, Seyed Alireza Hassani, Salva Ardid, Paul Tiesinga, and Thilo Womelsdorf. Feature-specific prediction errors and surprise across macaque fronto-striatal circuits. *Nature communications*, 10(1):176, 2019.

[4] Kimberly L Stachenfeld, Matthew M Botvinick, and Samuel J Gershman. The hippocampus as a predictive map. *Nature neuroscience*, 20(11):1643, 2017.

[5] Ida Momennejad, Evan M Russek, Jin H Cheong, Matthew M Botvinick, Nathaniel Douglass Daw, and Samuel J Gershman. The successor representation in human reinforcement learning. *Nature Human Behaviour*, 1(9):680, 2017.