Prioritized Sweeping Neural DynaQ with Multiple Predecessors, and Hippocampal Replays

Lise Aubin, Mehdi Khamassi, and Benoît Girard

Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique (ISIR), F-75005 Paris, France
benoit.girard@sorbonne-universite.fr

Abstract. During sleep and awake rest, the hippocampus replays sequences of place cells that have been activated during prior experiences. These have been interpreted as a memory consolidation process, but recent results suggest a possible interpretation in terms of reinforcement learning. The *Dyna* reinforcement learning algorithms use off-line replays to improve learning. Under limited replay budget, a *prioritized sweeping* approach, which requires a model of the transitions to the predecessors, can be used to improve performance. We investigate whether such algorithms can explain the experimentally observed replays. We propose a neural network version of prioritized sweeping Q-learning, for which we developed a growing multiple expert algorithm, able to cope with multiple predecessors. The resulting architecture is able to improve the learning of simulated agents confronted to a navigation task. We predict that, in animals, learning the world model should occur during rest periods, and that the corresponding replays should be shuffled.

Keywords: Reinforcement Learning, Replays, DynaQ, Prioritized Sweeping, Neural Networks, Hippocampus, Navigation

1 Introduction

The hippocampus hosts a population of cells responsive for the current position of the animal within the environment, the place cells (PCs), a key component of the brain navigation system [1]. Since the seminal work of [2], it has been shown that PCs are reactivated during sleep – obviously without any locomotion – and that these reactivations are functionally linked with the improvement of the learning performance of a navigation task [3]. Similar reactivations have been observed in the awake state [4], while the animal is immobile, either consuming food at a reward site, waiting at the departure site for the beginning of the next trial or stopped at a decision point. These reactivations contain sequences of PCs' activations experienced in the awake state (forward reactivations) [5], sequences played in the reverse order (backward reactivations) [4], and sometimes never experienced sequences (resulting from the concatenation of experienced sequences) [6]. These reactivations have been interpreted in the light of the

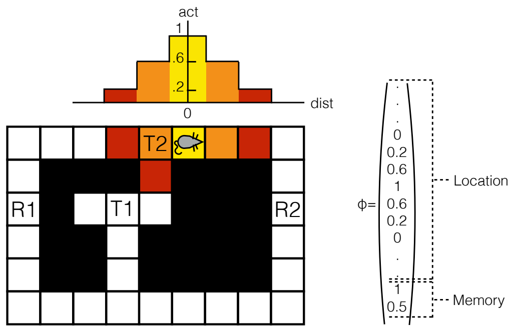

Fig. 1. Model of the rat experiment used in [6]. The maze is discretized into 32 positions (squares). The agent can use 4 discrete actions (N,E,S,W). The input state ϕ is the concatenation of 32 location components and two reward memory components. The location part of ϕ represents the activation of 32 place cells co-located with the maze discrete positions, their activity *act* depends on the Manhattan distance of the agent to the cell. All figures by Aubin & Girard, 2018; available at https://doi.org/10.6084/m9.figshare.5822112.v2 under a CC-BY4.0 license.

memory consolidation theory [7]: they would have the role of copying volatile hippocampal memories into the cortex [8] for reorganization and longer-term storage [9]. Some recent results have however shown that these reactivations also have a causal effect on reinforcement learning processes [3, 10].

A number of reinforcement learning (RL) algorithms make use of reactivations of their inputs, reminiscent of hippocampal reactivations, that are thus candidates to explain this phenomenon [11]. Among them, the Dyna family of algorithms [12] is of special interest because it was specifically designed to make the best possible use of alternation between on-line and off-line learning phases (i.e. phases during which the agent acts in the real world or in simulation). We concentrate here on the Q-learning version of Dyna (Dyna-Q). When operating on-line, Dyna-Q is indistinguishable from the original *model-free* Q-learning algorithm: it computes reward prediction error signals, and uses them to update the estimated values of the (state, action) couples, $Q(s, a)$. In its original version [12], when off-line, the Dyna algorithm reactivates randomly chosen quadruplets composed of an initial state, a chosen action, and the predicted resulting state and reward (produced by a learned world-model, this phase being thus *model-based*), in order to refine the on-line estimated values. However, when the number of reactivations is under a strict budget constraint, it is more efficient to select those that will provide more information: those that effectively generated a large reward prediction error in the last on-line phase, and those that are predicted to do so by the world model, a principle called *prioritized sweeping* [13, 14].

We are here interested in mimicking the process by which the basal ganglia, which is central for RL processes [15], can use the state representations of the world that are provided by the hippocampus. The manipulated state descriptor will thus be a population activity vector, and we will represent the Q-values and the world model with neural network approximators [16].

In the following, we describe the rat experimental setup proposed in [6], and how we simulated it. In this task, a state can have multiple predecessor states resulting from the execution of a single action, we thus present a modified Dyna-Q learning algorithm, with a special stress on the neural-network algorithm we designed to learn to approximate binary relations (not restricted to functions) with a *growing* approach: GALMO for Growing Algorithm to Learn Multiple Outputs. Our results successively illustrate three main points. First, because of interferences between consecutively observed states during maze experience, themselves due to the use of a neural-network function approximator, the world model had to be learned with shuffled states during off-line replay. Second, GALMO allows to efficiently solve the multiple predecessor problem. Third, the resulting system, when faced with a training schedule similar to [6], generates a lot of disordered state replays, but also a non-negligible set of varied backward and forward replay sequences, without explicitly storing and replaying sequences.

2 Methods

2.1 Experimental task

We aim at modeling the navigation task used in [6]: two successive T-mazes (T1 and T2 on **Fig. 1**), with lateral return corridors. The left and right rewarding sites deliver food pellets with different flavors. The training involves daily changing contingencies, forcing rats to adapt their choice to turn either left or right at the final choice (T2) based on the recent history of reward. These contingencies are: 1) always turn right, while the left side of the maze is blocked; 2) always turn left, while the right side of the maze is blocked; 3) always turn right; 4) always turn left; 5) alternate between left and right on a lap-by-lap basis.

Rats attempting to run backward on the maze were physically prevented to do so by the experimenter. They had forty trials the first day to learn task 1, and forty trials the second day to learn task 2. Then, depending on their individual learning speed, rats had between seventeen and twenty days to learn task 3, 4 and 5 (a single condition being presented each day). Once they reached at least 80% success rate on all tasks, rats were implanted with electrodes; after recovery, recording sessions during task performance lasted for six days.

During the six recording sessions, the reward contingency was changed approximately midway through the session and hippocampal replays were analyzed when rats paused at reward locations. Original analyses of replayed sequences [6] revealed that: during same-side replays (i.e., replays representing sequences of previously visited locations on the same arm of the maze as the current rat position) forward and backward replays started from the current position; during opposite-side replays (i.e., representing locations on the opposite arm of the maze) forward replays occurred mainly on the segment leading up to reward sites, and backward replays covered trajectories ending near reward sites. In general, the replay content did not seem to only reflect recently experienced trajectories, since trajectories experienced 10 to 15 minutes before were replayed as well. Indeed, there were more opposite-side replays during task 3 and 4 than during the alternation task. Finally, among all replays, a few were shortcuts never experienced before which crossed a straight path on the top or bottom of the maze between the reward sites.

2.2 Simulation

We have reproduced the T-maze configuration with a discrete environment composed of 32 squares (**Fig. 1, left**), each of them represents a 10×10 cm area. States are represented by a vector ϕ, concatenating place cells activity and a memory of past rewards (**Fig. 1, right**). The modeled place cells are centered on the discrete positions, their activity (color-coded on **Fig. 1**) decreases with the Manhattan distance between the simulated rat position to the position they encode (top of **Fig. 1**). When a path is blocked (contingencies 1 and 2), the activity field does not expand beyond walls and will thus shrink, as is the case of real place cells [17]. To represent the temporal dimension, which is essential during the alternation task, we have added two more components in the state's vector representation (**Fig. 1, right**): the left side reward memory (L) and the right side reward memory (R). They take a value of 1 if the last reward was obtained on that side, 0.5 if the penultimate reward was on that side, and 0 if that side has not been rewarded during the last two reward events. Therefore, after two successful laps, the task at hand can be identified by the agent based on the value of this memory (**Tab. 1**). This ability to remember the side of the rewards is supposed to be anchored both on the different position and flavor cues that characterize each side. Since it has been shown that, beyond purely spatial information, the hippocampus contains contextual information important for the task at hand [18], we hypothesize that this memory is also encoded within the hippocampus, along with the estimation of the agent's current position.

The agent can choose between four actions: North, South, East and West. As in the real experiment, the agent cannot run backward.

Table 1. State of the L and R memory components of ϕ and corresponding meaning in terms of task at hand, after two successful laps.

L	R	Task identification (after 2 laps)
1	0	Always turn right (Tasks 1 & 3)
0	1	Always turn left (Tasks 2 & 4)
0.5	1	Alternation (Task 5), go left next time
1	0.5	Alternation (Task 5), go right next time

2.3 Neural DynaQ with a prioritized sweeping algorithm

Our algorithm is based on a Dyna architecture [12] which means that, as in model-based architectures, we need to learn a world model composed of a reward and a transition model [19]. In order to implement *prioritized sweeping* [13, 14], the transition model must be designed so as to allow the prediction of the predecessors of a state s given an action a, because it will be needed to back-propagate the reward prediction computed in state s to its predecessors. Hence, our architecture is composed of two distinct parts: one dedicated to learning the world model, and the other one to learning the Q-values.

Algorithm 1 LearnWM: learn the world model

```
collect $\mathcal{S}$ // a set of $(\phi^t, \phi^{t-1}, a, r)$ quadruplets
for $k \in \{N, S, E, W\}$ do
    $\mathcal{S}_P^k \leftarrow \{(\phi^t, \phi^{t-1}) : (\phi^t, \phi^{t-1}, a, r) \in \mathcal{S}$ and $a = k\}$
    $\mathcal{S}_R^k \leftarrow \{(\phi^t, r) : (\phi^t, \phi^{t-1}, a, r) \in \mathcal{S}$ and $a = k\}$
    for $f \in \{P, R\}$ do
        // P,R: Predecessor and Reward types of networks
        $\mathcal{N}_f^k \leftarrow$ null // list of networks (outputs)
        $\mathcal{G}_f^k \leftarrow$ null // list of networks (gates)
        create $N_{new}^k$ ; append $N_{new}^k$ to $\mathcal{N}_f^k$
        create $G_{new}^k$ ; append $G_{new}^k$ to $\mathcal{G}_f^k$
        GALMO($\mathcal{S}_f^k$, $\mathcal{N}_f^k$, $\mathcal{G}_f^k$) // refer to Algo 2 for this specific training procedure
    end for
end for
```

Learning the world model. Two sets of neural networks compose the world model. Four reward networks N_R^a, one for each action a, learn the association between (state, action) couples and rewards ($N_R^a : s \rightarrow r(s, a)$). Four other networks N_P^a learn the states for which a transition to a given state s is produced after execution of action a, i.e., the predecessors of s ($N_P^a : s \rightarrow \{s'\}$).

Owing to the nature of the task (navigation constrained by corridors) and the states' representation, the data that must be learned are not independent. Indeed, successive state vectors are very similar due to the overlap between place-fields, and are always encountered in the same order during tasks execution (because the agent always performs the same stereotyped trajectories along the different corridors). However, it is well known that the training of a neural network is guaranteed to converge only if there is no correlation in the sequence of samples submitted during learning, a condition that is often not respected when performing on-line reinforcement learning [20]. We indeed observed that in the task at hand, despite its simplicity, it was necessary to store the successive observations and to train the world model off-line with a shuffled presentation of the training samples (for the general scheme of the off-line training, see **Algo. 1**). For that reason, we created a dataset $\mathcal{S}$ compiling all transitions, i.e (ϕ^t, ϕ^{t-1}, a, r) quadruplets from all tasks. When there is no predecessor of ϕ^t by action

a (as can be the case when this action would require to come through a wall), the transition is represented as (ϕ^t, $\mathbf{0}$, a, r): those "null" transitions allow N_P^a networks to represent the fact that the transition does not exist.

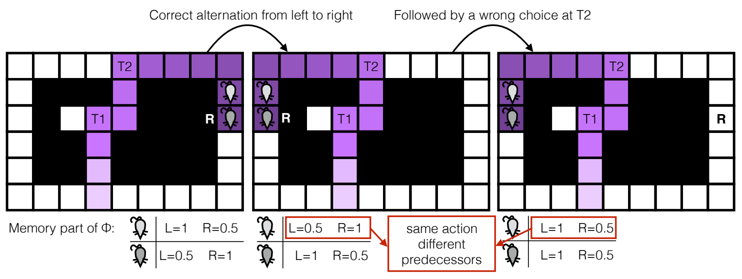
Correct alternation from left to right
Followed by a wrong choice at T2
T2
T1
R
Memory part of Φ:
L=1 R=0.5
L=0.5 R=1
L=0.5 R=1
L=1 R=0.5
same action different predecessors
L=1 R=0.5
L=1 R=0.5

Fig. 2. Example of multiple predecessors in the alternation task. The agent first correctly goes to the right (left). It then goes to the left (middle) where, at the reward site, its predecessor state has a $(L = 0.5, R = 1)$ memory component. It then makes a wrong decision and goes to the left again (right), but is not rewarded: at this last position, the location component of the predecessor state (white mouse) is identical but the memory component is different $(L = 1, R = 0.5)$ from the previous lap. Violet gradient: past trajectory; white mouse: previous position; gray mouse: current position; white R: agent rewarded; black R: current position of the reward.

Despite its simplicity, the navigation task modeled here has some specificities: during task 5 (alternation), some states have more than one predecessor for a given action (see an example on **Fig. 2**), the algorithm must thus be capable of producing more than one output for the same input. To do that, we have created a growing type of algorithm inspired by *mixture of expert* algorithms [21] (which we call here the *GALMO* algorithm, see **Algo. 2**), based on the following principles:

- The algorithm should allow the creation of multiple N_i networks (if needed) so that a single input can generate multiple outputs. Each of these network is coupled with a gating network G_i, used after training to know if the output of N_i has to be taken into account when a given sample is presented.
- When a sample is presented, the algorithm should only train the N_i network that generates the minimal error (to enforce network specialization), and remember this training event by training G_i to produce 1 and the other $G_{k\neq i}$ to produce 0.
- The algorithm should track the statistics of the minimal training errors of each sample during an epoch, so as to detect outliers (samples whose error is much higher than the others'). GALMO assumes that these outliers are caused by inputs who should predict multiple outputs and are stuck in predicting the barycenter of the expected outputs. A sample is considered an outlier when its error is larger than a threshold θ, equal to the median of the current error distribution, plus w times the amplitude of the third quartile

$(Q3 - median)$. When such a detection occurs, a new network is created on the fly, based on a copy of the network that produced the minimal error for the sample. The new network is then trained once on the sample at hand.

Algorithm 2 GALMO: Growing algorithm to learn multiple outputs

```
INPUT: $\mathcal{S}$, $\mathcal{N}$, $\mathcal{G}$
OUTPUT: $\mathcal{N}$, $\mathcal{G}$
// $\mathcal{S} = \langle (in_0, out_0), ..., (in_n, out_n) \rangle$ : list of samples
// $\mathcal{N} = \langle N_0 \rangle$ : lists of neural networks (outputs)
// $\mathcal{G} = \langle G_0 \rangle$ : lists of neural networks (gates)
$\theta \leftarrow +\infty$
for nbepoch $\in \{1, maxepoch\}$ do
  $\mathcal{M} \leftarrow$ null // $\mathcal{M}$ is a list of the minimal error per sample
  for each (in,out)$\in \mathcal{S}$ do
    $\mathcal{E} \leftarrow$ null // $\mathcal{E}$ is a list of errors for a sample
    for each $N \in \mathcal{N}$ do
      append $\|N(in) - out\|_{L_1}$ to $\mathcal{E}$
    end for
    if min($\mathcal{E}$) $< \theta$ then
      backprop($N_{argmin(\mathcal{E})}, in, out$)
      backprop($G_{argmin(\mathcal{E})}, in, 1$)
      for each $G \in \mathcal{G}$ with $G \neq G_{argmin(\mathcal{E})}$ do
        backprop($G, in, 0$)
      end for
    else
      create $N_{new}$; append $N_{new}$ to $\mathcal{N}$
      $N_{new} \leftarrow$ copy($N_{argmin(\mathcal{E})}$)
      backprop($N_{new}$, $input$ =in, $target$ =out)
      create $G_{new}$; append $G_{new}$ to $\mathcal{G}$
      backprop($G_{new}, in, 1$)
    end if
  end for
  $\theta \leftarrow median(\mathcal{M}) + w * (Q3(\mathcal{M}) - median(\mathcal{M}))$
end for
```

In principle, the algorithm could be modified to limit the maximal number of created networks, or to remove the networks that are not used anymore, but these additions were not necessary here.

Neural Dyna-Q. The second part of the algorithm works as a classical neural network-based Dyna-Q [16] with *prioritized sweeping* [13, 14]. As in [16], the Q-values are represented by four 2-layer feedforward neural networks N_Q^a (one per action). During on-line phases, the agent makes decisions that drive its movements within the maze, and stores the samples in a priority queue, their priority is the absolute value of the reward prediction error, i.e., $|\delta|$. Every time the agent receives a reward, similarly to rats, it stops and replays are simulated with a budget B (**Algo. 3**): the samples with the highest priority are replayed

Algorithm 3 Neural Dyna-Q with *prioritized sweeping* & multiple predecessors

```
INPUT: $\phi^{t=0}$, $\mathcal{N_P}$, $\mathcal{G_P}$, $\mathcal{N_R}$, $\mathcal{G_R}$
OUTPUT: $N_Q^{a \in \{N,S,E,W\}}$
PQueue $\leftarrow$ {} // PQueue: empty priority queue
nbTrials $\leftarrow$ 0
repeat
  a $\leftarrow$ softmax($N_Q(\phi^t)$)
  take action a, receive r, $\phi^{t+1}$
  backprop($N_Q^a$, $input = \phi^t$, $target = r + \gamma max_a(N_Q(\phi^{t+1}))$
  Put $\phi^t$ in PQueue with priority $|N_Q^a(\phi^t) - (r + \gamma max_a(N_Q(\phi^{t+1})))|$
  if r> 0 then
    nbReplays $\leftarrow$ 0
    Pr = $\langle\rangle$ // empty list of predecessors
    repeat
      $\phi \leftarrow$ pop(PQueue)
      for each $G_P \in \mathcal{G}_P$ do
        if $G_P(\phi) > 0$ then
          $k \leftarrow$ index($G_P$)
          append $N_P^k(\phi)$ to Pr
        end if
      end for
      for each $p \in$ Pr s.t norm(p) $> \epsilon$ do
        for each $a \in \{N, S, E, W\}$ do
          backprop($N_Q^a$, $input$ = p, $target = N_R^a(p) + \gamma max_a(N_Q^a(\phi))$)
          Put p in PQueue with priority $|N_Q^a(p) - (N_R^a(p) + \gamma max_a(N_Q^a(\phi)))|$
          nbReplays $\leftarrow$ nbReplays + 1
        end for
      end for
    until PQueue empty OR nbReplays $\geq$ B
  end if
  $\phi^t \leftarrow \phi^{t+1}$
  nbTrials $\leftarrow$ nbTrials +1
until nbTrials = maxNbTrials
```

first, their potential predecessors are then estimated and placed in the queue with their respective priorities, and so on until the replay budget is exhausted.

The various parameters used in the simulations are summarized in **Tab. 2**.

3 Results

3.1 Learning the world model

Because of correlations in sample sequences, the world model is learned off-line: the samples are presented in random order, so as to break temporal correlations. We illustrate this necessity with the learning of the reward networks N_R: when trained on-line (**Fig. 3, left**), the reward networks make a lot of erroneous pre-

Table 2. Parameter values.

value	parameter
4000	*maxepch*: number of epoch replays to train the world model
3	w: gain of the outlier detector threshold in GALMO
20	B: replay budget per stop at reward sites
2	number of layers in N_P, N_R and N_Q
10, 16, 26	size of the hidden layers in N_Q, N_R and N_P (respectively)
±0.05, ±0.0045, ±0.1	weight initialization bound in N_Q, N_R and N_P (resp.)
0.5, 0.1, 0.1	learning rate in N_Q, N_R and N_P (resp.)
0.9, 1, 1	sigmoid slope in N_P, N_R and N_Q (resp.) (hidden layer)
0.5, 0.4, 0.4	sigmoid slope in N_P, N_R and N_Q (resp.) (output layer)

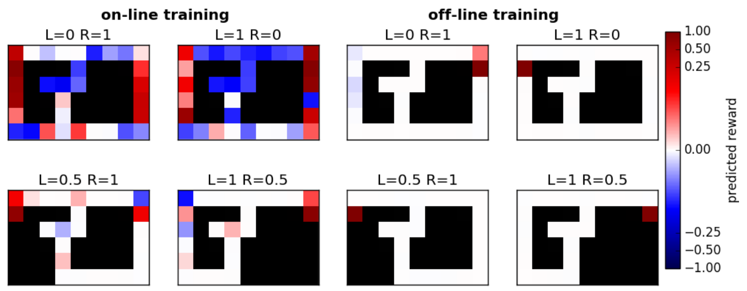

Fig. 3. Reward predictions are inaccurate when the model is trained on-line (Left panel) and accurate when it is trained off-line (Right panel). L, R: memory configuration. Note the use of a logarithmic scale, so as to make visible errors of small amplitude.

dictions for each possible task, while when trained off-line with samples presented in randomized order, the predictions are correct (**Fig. 3, right**).

With a single set of N_P networks, the error of the whole set of states decreases steadily with learning, except for four states which have multiple predecessors (**Fig. 4, top left**). With the GALMO algorithm, when the error of these states reaches the threshold θ (in red on **Fig. 4, top right**), networks are duplicated and specialized for each of the possible predecessors. We repeated the experiment 10 times. It always converged, with a number of final networks comprised between 2 and 5 (3 times 2 networks, 1 time 3, 5 times 4, and 1 time 5).

3.2 Reinforcement learning with multiple predecessors

We compare the efficiency of the Dyna-Q model we developed with the corresponding Q-learning (i.e. the same architecture without replays with the world model). As expected, Q-learning is able to learn the task, measured here with the proportion of erroneous choices at the decision point T2 (**Fig. 4, bottom left**). On average it does not fully converge before 1000 epochs of training, the Dyna-Q learns much faster, thanks to the replay mechanism (**Fig. 4, bottom right**), converging on average after 200 trials.

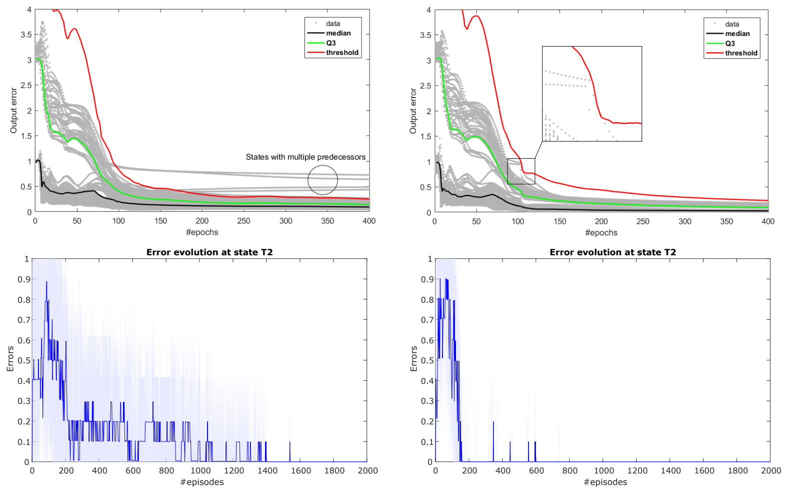

Fig. 4. Top: Learning error dynamics without (left) and with (right) GALMO. Errors of all samples (gray) during epochs of training. GALMO allows for the creation of multiple prediction networks to handle the states where multiple outputs have to be generated. **Bottom: Learning without (left) and with (right) replays.** Evolution of the proportion of decision errors at point T2 during the alternation task. Blue: 10 run average, light blue: standard deviation.

3.3 Preliminary analysis of generated replays

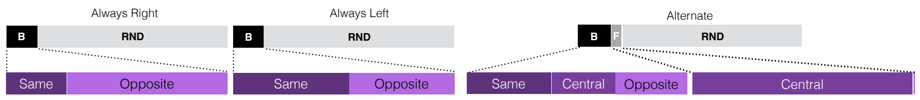

Fig. 5. Type of replays. B: backward, F: forward, RND: random.

We analyze a posteriori the state reactivations caused by the prioritized sweeping DynaQ algorithm in always turn right, always turn left and alternate tasks. Prioritized sweeping does not rely on explicit replay of sequences, but it however favors them. We considered sequences or replays implying three or more consecutive steps; with 128 possible states, a 3-state sequence has a chance level of 0.01% of being produced by a uniform random selection process. We observed in all cases (Fig. 5) that a bit more than 80% of the state reactivations did not correspond to actual sequences. Most of the sequences are backward, except for the alternate task, which also generated 4.5% of forward ones. As in [6], we classified these sequences as being on the same side as the current agent location, on the opposite side, or in the central part of the maze. There is no clear pattern here, except that central reactivations were observed in the alternate task only.

4 Discussion

We proposed a new neural network architecture (GALMO) designed to associate multiple outputs to a single input, based on the multiple expert principle [21]. We implemented a neural version of the DynaQ algorithm [16], using the prioritized sweeping principle [13, 14], using GALMO to learn the world model. This was necessary because the evaluation task, adapted from [6], contained some states that have multiple predecessor.

We showed that this system is able to learn the multiple predecessors cases, and to solve the task faster than the corresponding Q-learning system (i.e., without replays). This required learning the world-model off-line, with data presented in shuffled order, so as to break the sequential correlations between them, which prevented the convergence of the learning process. A neuroscience prediction derives from this result (independently from the use of GALMO): if the learning principles of the rat brain are similar to those of the gradient descent for artificial neural network, then the world model has to be learned off-line, which would be compatible with non-sequential hippocampal replays. Besides, the part of the DynaQ algorithm that uses the world model to update the Q-values predicts a majority of non-sequential replays, but also 15 to 20% of sequential reactivations, both backward and forward.

Concerning GALMO, it has been tested with a quite limited set of data, and should thus be evaluated against larger sets in future work. In our specific case, the reward networks N_R^a did not require the use of GALMO; a single network could learn the full $(s, a) \rightarrow r$ mapping as rewards were deterministic. But should they be stochastic, GALMO could be used to learn the multiple possible outcomes. Note that, while it has been developed in order to learn a predecessor model in a DynaQ architecture, GALMO is much more general, and would in principle be able to learn any one-to-many mapping. Finally, in the model-based and dyna reinforcement learning contexts, having multiple predecessors or successors is not an exceptional situation, especially in a robotic paradigm. The proposed approach is thus of interest beyond the task used here.

Acknowledgements

The authors would like to thank Olivier Sigaud for fruitful discussions. This work has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 640891 (DREAM Project). This work was performed within the Labex SMART (ANR-11-LABX-65) supported by French state funds managed by the ANR within the Investissements d'Avenir programme under reference ANR-11-IDEX-0004-02.

References

1. O'Keefe, J., Dostrovsky, J.: The hippocampus as a spatial map. preliminary evidence from unit activity in the freely-moving rat. Brain research **34**(1) (1971) 171–175

2. Wilson, M.A., McNaughton, B.L., et al.: Reactivation of hippocampal ensemble memories during sleep. Science **265**(5172) (1994) 676–679
3. Girardeau, G., Benchenane, K., Wiener, S.I., Buzsáki, G., Zugaro, M.B.: Selective suppression of hippocampal ripples impairs spatial memory. Nature neuroscience **12**(10) (2009) 1222–1223
4. Foster, D.J., Wilson, M.a.: Reverse replay of behavioural sequences in hippocampal place cells during the awake state. Nature **440**(7084) (2006) 680–3
5. Lee, A.K., Wilson, M.A.: Memory of Sequential Experience in the Hippocampus during Slow Wave Sleep. Neuron **36**(6) (2002) 1183–1194
6. Gupta, A.S., van der Meer, M.A.A., Touretzky, D.S., Redish, A.D.: Hippocampal Replay Is Not a Simple Function of Experience. Neuron **65**(5) (2010) 695–705
7. Chen, Z., Wilson, M.A.: Deciphering neural codes of memory during sleep. Trends in Neurosciences (2017)
8. Peyrache, A., Khamassi, M., Benchenane, K., Wiener, S.I., Battaglia, F.P.: Replay of rule-learning related neural patterns in the prefrontal cortex during sleep. Nature Neuroscience **12**(7) (2009) 919–926
9. McClelland, J.L., McNaughton, B.L., O'reilly, R.C.: Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. Psychological review **102**(3) (1995) 419
10. De Lavilléon, G., Lacroix, M.M., Rondi-Reig, L., Benchenane, K.: Explicit memory creation during sleep demonstrates a causal role of place cells in navigation. Nature neuroscience **18**(4) (2015) 493–495
11. Cazé, R., Khamassi, M., Aubin, L., Girard, B.: Hippocampal replays under the scrutiny of reinforcement learning models. submitted (2018)
12. Sutton, R.S.: Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In: Proceedings of the seventh international conference on machine learning. (1990) 216–224
13. Moore, A.W., Atkeson, C.G.: Prioritized sweeping: Reinforcement learning with less data and less time. Machine learning **13**(1) (1993) 103–130
14. Peng, J., Williams, R.J.: Efficient learning and planning within the dyna framework. Adaptive Behavior **1**(4) (1993) 437–454
15. Khamassi, M., Lacheze, L., Girard, B., Berthoz, A., Guillot, A.: Actor-critic models of reinforcement learning in the basal ganglia: from natural to arificial rats. Adaptive Behavior **13** (2005) 131–148
16. Lin, L.H.: Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine learning **8**(3/4) (1992) 69–97
17. Paz-Villagrán, V., Save, E., Poucet, B.: Independent coding of connected environments by place cells. European Journal of Neuroscience **20**(5) (2004) 1379–1390
18. Eichenbaum, H.: Prefrontal–hippocampal interactions in episodic memory. Nature Reviews Neuroscience **18**(9) (2017) 547
19. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. Cambridge, MA: MIT Press (1998)
20. Tsitsiklis, J.N., Van Roy, B.: Analysis of temporal-diffference learning with function approximation. In: Advances in neural information processing systems. (1997) 1075–1081
21. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E.: Adaptive mixtures of local experts. Neural computation **3**(1) (1991) 79–87