Prioritized Sweeping Neural DynaQ: Boost Learning by Dreaming

Alexander Osiik
alexander.osiik@student.uni-luebeck.de
Seminar Cyber-Physical Systems (WS 2019/20)
Institute of Computer Engineering, University of Lübeck

January 4, 2020

Abstract

Hier beschreiben welche Hintergrnde und Analogien aus der Biologie Reinforcement Learning hat.
State of the Art.
Was Zielsetzung im Original-Paper war, wie sie erreicht wurde und wie es im Projekt umgesetzt wurde.

1 Introduction

The hippocampus is the main memory of our brain and the switch point between short-term and long-term memory. O'Keefe and Dostrovsky [1971] observed in several rat experiments that if there is a disturbance in this area, no new information can be stored in the brain. Regarding animals, the hippocampus is known to be important for navigation since the discovery of place cells, which signal the current allocentric location of the animal in the environment [Maguire et al., 2006]. Interaction within the environment leads to activation these place cells. They are engaged during the consolidation of spatial memories. However, it has been observed that these activations also occur during rest or sleep, at a significally faster pace. This reactivation can be seen as a reprocessing of the experience gained during the day, something that is usually the case when dreaming [Caz et al., 2018].
Aubin et al. [2018] presented an approach to convert the reactivation of the hippocamus' place cells into a reinforcement learning model. O'Keefe and Dostrovsky [1971] postulated that the hippocampus functions as a spatial map, where single hippocampal neurons increased their firing rate whenever a rat traversed a particular region of an environment, as concluded by Nakazawa et al. [2004].

1.1 Experimental setup

Aubin et al. [2018] set up a experimental task, which was derived and slightly modified from Gupta et al. [2010]. The environment consisted of two succesive T-mazes with lateral return corridors and rewarding food pellets on each side, see Figure 1. A rat was placed in the maze, and trained to make a decision

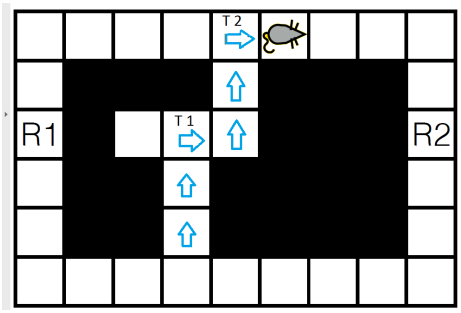

Figure 1: Discretized experimental setup. The agent has to make a decision at points T1 and T2, in which the dicision at T2 leads to one of the rewarding sites. [Aubin et al., 2018]

at position T2, with the objective of getting the reward on the left or right hand side based on the task pursued at the moment. The tasks were 1) always turn right, 2) always turn left, 3) alternate between left and right. At reward locations, the rat's hippocampal replays were analyzed. It has been shown that the rats reflected recently experienced trajectories, and, in addition, also those that occurred a longer time ago.
To mathematiacally model and explain the hippocampal replay phenomenon, algorithms from the Dyna family of algorithms were used. Dyna is an integrated architecture for learning, planning and reacting, proposed by Sutton [1991], see Figure 2. The Dyna idea is a trivial approach that planning is equivalent to trying out multiple things in the own mind, under the condition that a certain internal model of the problem exists. Ultimately, this architecture was chosen because it is designed to make the best possible use of alternation between on-line and off-line learning phases [Sutton, 1991]. Aubin et al. [2018] concentrated on the Q-learning version of Dyna (Dyna-Q) extended by prioritized sweeping, by that optimizing the choice of reactivated cells.

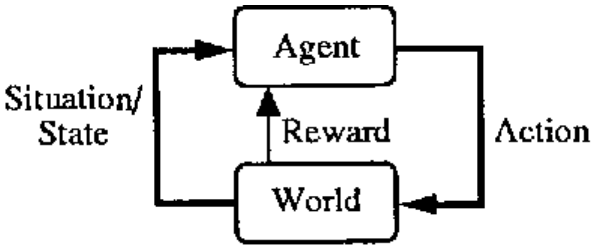

Figure 2: The Problem Formulation Used in Dyna. The agent's objective is to maximize the total reward it receives over time. [Sutton, 1991]

2 Reinforcement Learning

To further understand the underlying methods it is important to know the general mathematical rules and machine learning approaches. In the following,

the terminology aswell as mathematical background of the research subject is briefly summarized and explained.

2.1 Markov Decision Problem

A Markov Decision Problem (MDP) is a model for problems, where an agent tries to interact with the environment in such way that the utmost reward is achieved. Specifically, the robot is moving (*transition*) through various *states*, having chosen a specific *action* in each state. Each *reward* is determined by initial state, the action and the following state. All transitions are not deterministic, but rather probabilistic, where each probability only depends on the current state and the current action. This memoryless property of stochastic processes is refered to as **Markov condition**. That way there has to be one initial state, but multiple end states are possible.
The main goal is to find a reward maximizing **policy**, by which the agent selects actions where the maximum reward can be expected. This policy is an optimal plan, or sequence of actions, from start to a goal [Klein and Abbeel, 2019].
A markov decision process consists of

- Set of states $S : \{s_1, s_2, \ldots, s_n\}$
- Set of actions $A : \{a_1, a_2, \ldots, a_n\}$
- Transition function, which is the probability of going from state s to state s' via action a: $T : S \times A \times S$
- Reward function $R : S \times A \times S \to \mathbb{R}$

The main goal is to find an optimal policy Π and exploration factor γ, where

- Π is the policy, where an optimal action is assigned to each state
- $\gamma \in [0, 1]$ is the discount factor. This factor determines the degree of exploration for the agent. For $\gamma = 0$, the agent will stick to the policy, and explot the current (possibly) optimal policy. For $\gamma = 1$, the agent will take into account the next states reward, leading to exploration behaviour. A value ¡1 is a contraint, which limits the maximal obtainable reward, leading to a reduction of cycles.

The **V-values** and **Q-values** are certain grading schemes used To solve MDPs. The values are the total reward the agent can expect if it performs the optimal, or maximally benefitting, action a in state s, and continues to act optimally thereafter:

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} T(s, a, s')[R(s, a, s') + \gamma V^*(s')] \tag{1}$$

As the values from equation 1 are hard do compute, a technique called **value iteration** is used. It is used do discretize the equation:

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} T(s, a, s')[R(s, a, s') + \gamma V^*(s')] \tag{2}$$

where k is the radius from the goal state to the agent. For example, regarding the manhattan norm in a 2D plane, the amount of steps the agent has left until

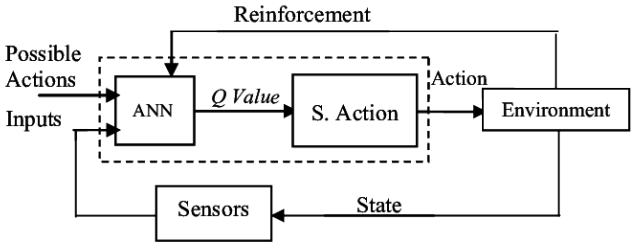

Figure 3: The structure of reinforcement learning based on an Artificial Neural Network [Hatem and Abdessemed, 2009]

it reaches an end state.
After that, **policy extraction** is performed. It is the assignment of an action to each state, maximizing the expected total reward:

$$\Pi^*(s) = \arg\max_{a \in A} \sum_{s' \in S} T(s, a, s')[R(s, a, s') + \gamma V^*(s')] \tag{3}$$

2.2 Q-Learning

Q-Learning is a model-free reinforcement learning algorithm which converges to optimal policy even if if the performed actions are suboptimal [Klein and Abbeel, 2019]. The letter **Q** stands for **quality** of a certain action in a given state, as the values represent the maximum discounted future reward when action a in state s is performed. Q-values are calculated similar to the value iteration in Equation 2:

$$Q^*(s, a) = \max_{a \in A} \sum_{s' \in S} T(s, a, s')[R(s, a, s') + \gamma V^*(s')] \tag{4}$$

The funcion $Q(s, a)$ can be estimated using Q-Learning, which is a form of **Temporal Difference Learning** (TD). TD means that the performs an update after every action based on a maximum reward estimate, and not only after receiving the reward.
Here, value $Q(s, a)$ is iteratively updated using the **Bellman Equation**:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a' \in A} Q(s', a') - Q(s, a)] \tag{5}$$

The Q-values are represented in a **Q-table** where each cell represents the highest achievable reward performing the state-action pair ($s \in S, a \in A$). This makes Q-learning suitable for small environments only, as an increase in states or actions increases the memory consumption quadratically.
The challenge is to tune the parameters α, γ such that the best possible policy is found in minimal time.

3 GALMO

TODO

- Brief summary of the proposed algorithm (GALMO)
- Why this algorithm? What are the benefits of dynamic DQN over simple Q-Learn?
- Present results

4 Project

As part of practical work, a simplified T-maze configuration with a discrete environment similar to the work of Aubin et al. [2018] has been reproduced. The programming language of choice was Python, as it offers many up-to-date machine learning libraries and allows a readable and understandable implementation.
For the successful completion of task 3 (reward sites alternate) the state space was extended by two components: the left side reward memory (L) and the right side reward memory (R). As proposed by Aubin et al. [2018], they take a value 1 if the last reward was obtained on that side, 0.5 if the penultimate reward was obtained on that side and 0 if that side has not been reward during the last 2 runs. Thus Q-Learning is also applicable in situations that actually depend on the past, by this violating the Markov condition.

TODO

- Umsetzung von Q-Learning in Python
- DQN
- Implementierung GALMO? prolly not, fam

5 Results

TODO
Compare own results and GALMO results

6 Conclusion

TODO

References

L. Aubin, M. Khamassi, and B. Girard. Prioritized sweeping neural dynaq with multiple predecessors, and hippocampal replays. In *Living Machines 2018*, LNAI, page TBA, Paris, France, 2018. URL https://hal.archives-ouvertes.fr/hal-01709275.

Romain Caz, Mehdi Khamassi, Lise Aubin, and Benot Girard. Hippocampal replays under the scrutiny of reinforcement learning models. *Journal of Neurophysiology*, 120, 10 2018. doi: 10.1152/jn.00145.2018.

Anoopum S. Gupta, Matthijs A.A. van der Meer, David S. Touretzky, and A. David Redish. Hippocampal replay is not a simple function of experience. *Neuron*, 65(5):695 – 705, 2010. ISSN 0896-6273. doi: https://doi.org/10.1016/j.neuron.2010.01.034. URL http://www.sciencedirect.com/science/article/pii/S0896627310000607.

Mezaache Hatem and Foudil Abdessemed. Simulation of the navigation of a mobile robot by the qlearning using artificial neuron networks. volume 547, 01 2009.

Dan Klein and Pieter Abbeel. Uc berkeley cs188 intro to ai, 2019. URL http://ai.berkeley.edu/.

Eleanor Maguire, Rory Nannery, and Hugo Spiers. Navigation around london by a taxi driver with bilateral hippocampal lesions. *Brain : a journal of neurology*, 129:2894–907, 12 2006. doi: 10.1093/brain/awl286.

Kazu Nakazawa, Thomas Mchugh, Matthew Wilson, and Susumu Tonegawa. Nmda receptors, place cells and hippocampal spatial memory. *Nature reviews. Neuroscience*, 5:361–72, 06 2004. doi: 10.1038/nrn1385.

J. O'Keefe and J. Dostrovsky. The hippocampus as a spatial map. preliminary evidence from unit activity in the freely-moving rat. *Brain Research*, 34(1):171 – 175, 1971. ISSN 0006-8993. doi: https://doi.org/10.1016/0006-8993(71)90358-1. URL http://www.sciencedirect.com/science/article/pii/0006899371903581.

Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160163, July 1991. ISSN 0163-5719. doi: 10.1145/122344.122377. URL https://doi.org/10.1145/122344.122377.