Prioritized Sweeping Neural DynaQ: Neural Networks over All?

Alexander Andreevič Osiik

Universität zu Lübeck

January 17th 2020

Introduction

- In recent years, **reinforcement learning** has gained a lot of popularity due to some spectacular successes

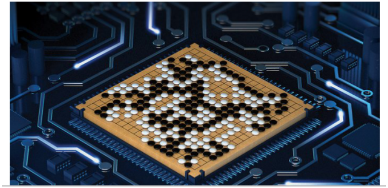
Introduction

- In recent years, **reinforcement learning** has gained a lot of popularity due to some spectacular successes

Künstliche Intelligenz: AlphaGo Zero übertrumpft AlphaGo ohne menschliches Vorwissen

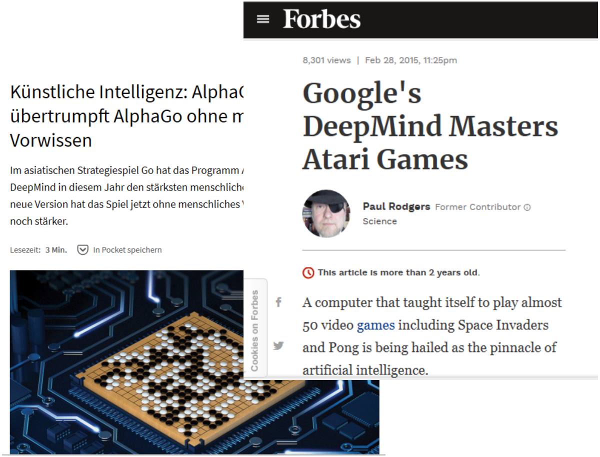
Im asiatischen Strategiespiel Go hat das Programm AlphaGo der Google-Tochter DeepMind in diesem Jahr den stärksten menschlichen Profispieler besiegt. Eine neue Version hat das Spiel jetzt ohne menschliches Vorwissen gelernt und spielt noch stärker.

Lesezeit: **3 Min.** In Pocket speichern 304

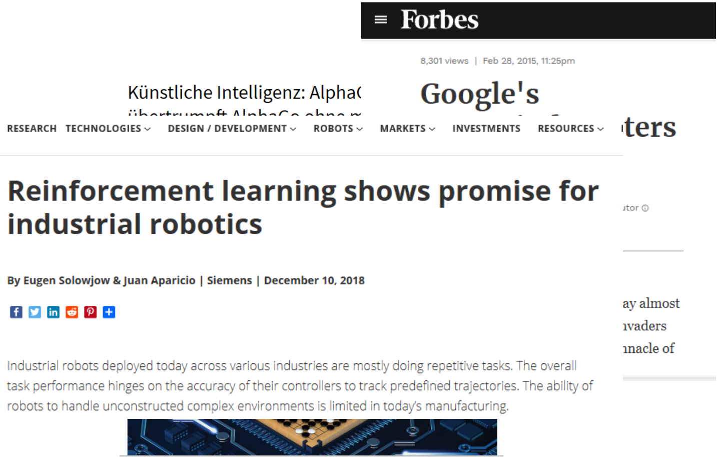


Introduction

- In recent years, **reinforcement learning** has gained a lot of popularity due to some spectacular successes



- In recent years, **reinforcement learning** has gained a lot of popularity due to some spectacular successes



Introduction

- **Reinforcement learning** (RL) is based on learning processes of biological systems

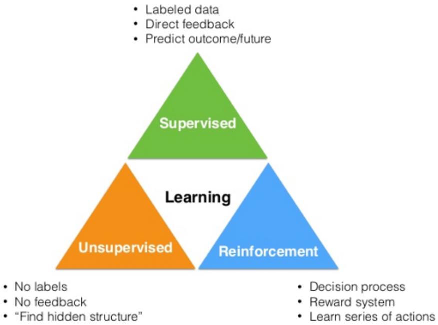


Figure: Categories of Machine Learning [Dishpande, 2016]

Introduction

- **Reinforcement learning** (RL) is based on learning processes of biological systems
- **Supervised learning**: create a model based on training set to classify input

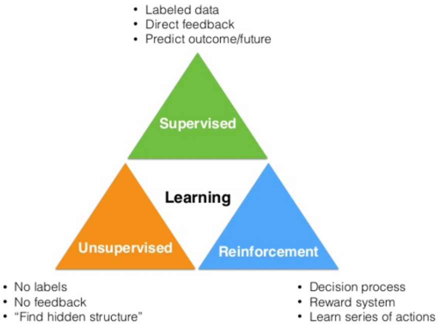


Figure: Categories of Machine Learning [Dishpande, 2016]

Introduction

- **Reinforcement learning** (RL) is based on learning processes of biological systems
- **Supervised learning**: create a model based on training set to classify input
- **Unsupervised learning**: sort data structures through cluster analysis

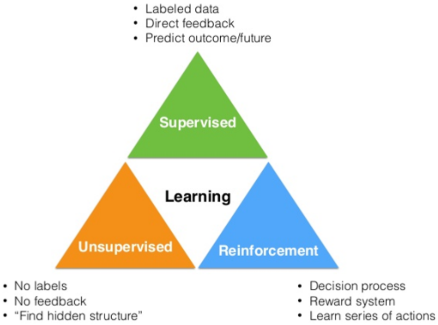


Figure: Categories of Machine Learning [Dishpande, 2016]

Limitations

Although RL can be effectively applied to some problems, there are some limitations:

Limitations

Although RL can be effectively applied to some problems, there are some limitations:

- Parameters affect speed of learning

Limitations

Although RL can be effectively applied to some problems, there are some limitations:

- Parameters affect speed of learning
- Reward function is required

Limitations

Although RL can be effectively applied to some problems, there are some limitations:

- Parameters affect speed of learning
- Reward function is required
- Storage of world models: computing-heavy and time-consuming for large dynamic environments

Limitations

Although RL can be effectively applied to some problems, there are some limitations:

- Parameters affect speed of learning
- Reward function is required
- Storage of world models: computing-heavy and time-consuming for large dynamic environments
- RL is still orders of magnitude above a practical level of sample efficiency [Irpan, 2018]

Limitations

Although RL can be effectively applied to some problems, there are some limitations:

- Parameters affect speed of learning
- Reward function is required
- Storage of world models: computing-heavy and time-consuming for large dynamic environments
- RL is still orders of magnitude above a practical level of sample efficiency [Irpan, 2018]

$\Longrightarrow$ The last two problems are tackled by Aubin et. al [Aubin et al., 2018]

Paper topic

Recent progress in reinforcement learning: Development of GALMO for Prioritized Sweeping Neural DynaQ Learning by [Aubin et al., 2018].

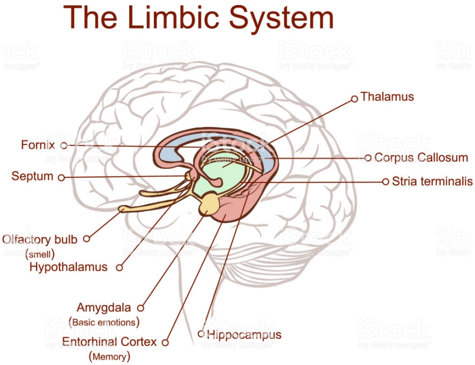
This solution promises to boost learning due to offline replays, a model of the *hippocampal replays* that occur in rodents.

GALMO architecture can also associate multiple outputs to a single input.

Outline

Biological background

The hippocampus is the main memory of the brain and the switch point between short-term and long-term memory.

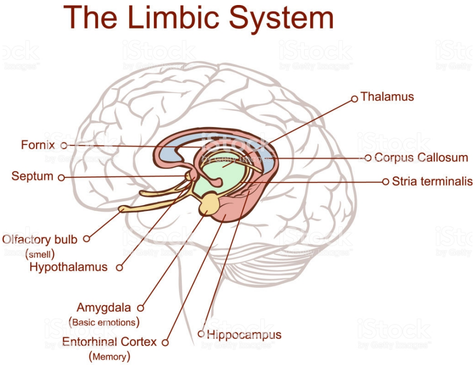


Biological background

The hippocampus is the main memory of the brain and the switch point between short-term and long-term memory.

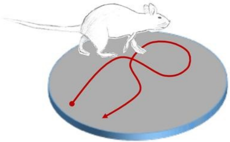
- Disturbance in this area leads to loss of the ability to store new information [O'Keefe and Dostrovsky, 1971], [Maguire et al., 2006]

$\Longrightarrow$ Learning no longer possible!



Biological background

[O'Keefe and Dostrovsky, 1971] discovered the presence of *place cells* in the hippocampus.

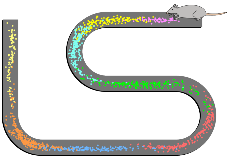


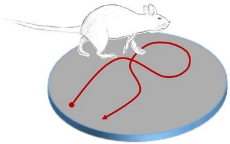
Biological background

[O'Keefe and Dostrovsky, 1971] discovered the presence of *place cells* in the hippocampus.

- Interaction with the environment leads to activation of these cells

$\Longrightarrow$ Postulation, that the hippocampus functions as a spatial map





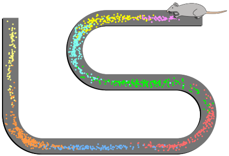
Biological background

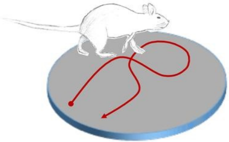
[O'Keefe and Dostrovsky, 1971] discovered the presence of *place cells* in the hippocampus.

- Interaction with the environment leads to activation of these cells

$\Longrightarrow$ Postulation, that the hippocampus functions as a spatial map

- It has also been observed that activations occur during rest or sleep, at significally faster pace





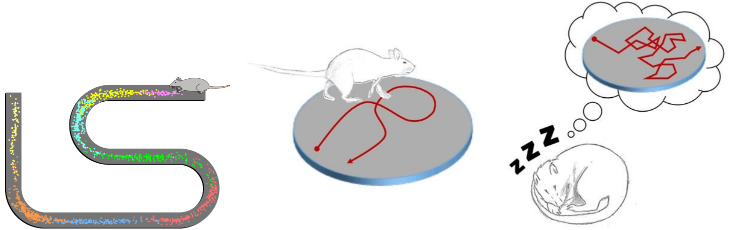
Biological background

[O'Keefe and Dostrovsky, 1971] discovered the presence of *place cells* in the hippocampus.

- Interaction with the environment leads to activation of these cells

$\Longrightarrow$ Postulation, that the hippocampus functions as a spatial map

- It has also been observed that activations occur during rest or sleep, at significally faster pace



Goal: Convert reactivation of place cells into a RL model

Outline

Markov Decision Process

MDP is a framework for modeling an agent's decision making in an environment

Markov Decision Process

MDP is a framework for modeling an agent's decision making in an environment

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

Markov Decision Process

MDP is a framework for modeling an agent's decision making in an environment

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

Markov Decision Process

MDP is a framework for modeling an agent's decision making in an environment

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

$\Longrightarrow$ Goal is to find policy Π, where an optimal action is assigned to each state

V-values and Q-values

MDP contains a *value function* V for its *states*. It is the expected reward for being in a state $s \in S$

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

$\gamma \in [0, 1]$: Exploration factor

V-values and Q-values

MDP contains a *value function* V for its *states*. It is the expected reward for being in a state $s \in S$

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

$\gamma \in [0,1]$: Exploration factor

V-values and Q-values

MDP contains a *value function* V for its *states*. It is the expected reward for being in a state $s \in S$

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

$\gamma \in [0,1]$: Exploration factor

What if there is no model of the environment (= no knowledge about states)?

V-values and Q-values

MDP contains a *value function* V for its *states*. It is the expected reward for being in a state $s \in S$

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

$\gamma \in [0,1]$: Exploration factor

What if there is no model of the environment (= no knowledge about states)?
$\implies$ Use the *action value function* Q instead

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

Q-learning

The function $Q(s,a)$ can be estimated using **Q-Learning**. The value $Q(s,a)$ is iteratively updated using the Bellman Equation:

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

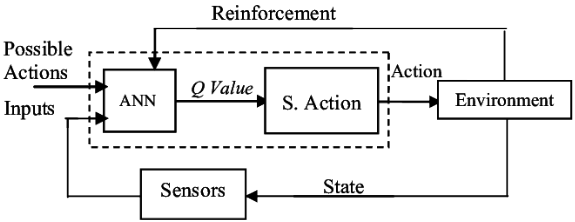
α : Learning rate

The values are stored in a **Q-table**, where each cell $(s \in S, a \in A)$ represents the highest achievable reward performing action a in state s.

Q-learning

The amount of time to explore and store each state in the Q-table would be incredibly large for large environments.

In **Deep Q-learning** (DQN), neural networks are used to approximate the Q-value function, where the state is the network's input and the Q-values of all actions are its output.



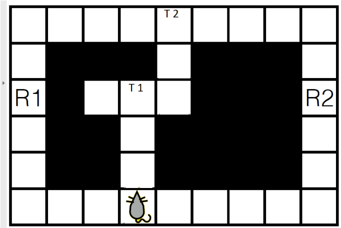
[Hatem and Abdessemed, 2009]

Outline

Experimental setup

[Aubin et al., 2018] set up an experimental task environment.

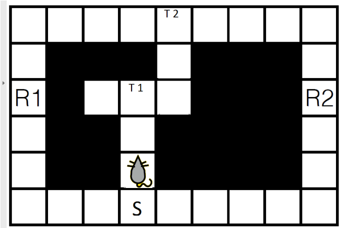
- Two successive T-mazes with return corridors
- Decision points T1 and T2
- Rewarding food pellets at each side



Experimental setup

[Aubin et al., 2018] set up an experimental task environment.

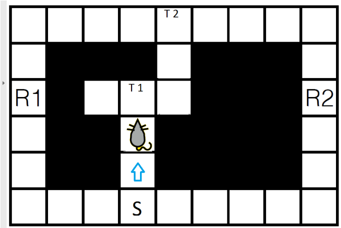
- Two successive T-mazes with return corridors
- Decision points T1 and T2
- Rewarding food pellets at each side



Experimental setup

[Aubin et al., 2018] set up an experimental task environment.

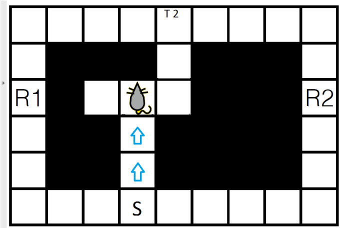
- Two successive T-mazes with return corridors
- Decision points T1 and T2
- Rewarding food pellets at each side



Experimental setup

[Aubin et al., 2018] set up an experimental task environment.

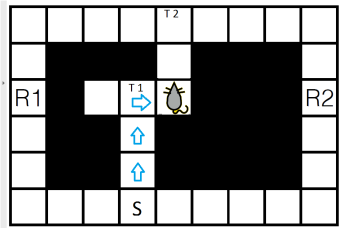
- Two successive T-mazes with return corridors
- Decision points T1 and T2
- Rewarding food pellets at each side



Experimental setup

[Aubin et al., 2018] set up an experimental task environment.

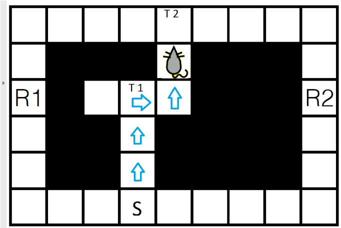
- Two successive T-mazes with return corridors
- Decision points T1 and T2
- Rewarding food pellets at each side



Experimental setup

[Aubin et al., 2018] set up an experimental task environment.

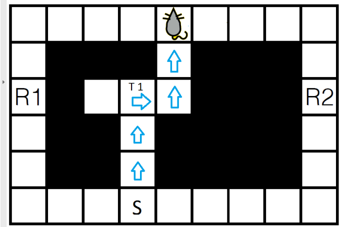
- Two successive T-mazes with return corridors
- Decision points T1 and T2
- Rewarding food pellets at each side



Experimental setup

[Aubin et al., 2018] set up an experimental task environment.

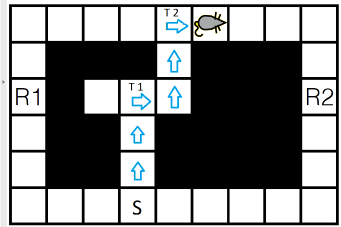
- Two successive T-mazes with return corridors
- Decision points T1 and T2
- Rewarding food pellets at each side



Experimental setup

[Aubin et al., 2018] set up an experimental task environment.

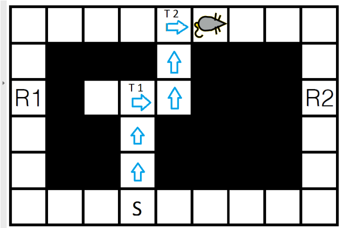
- Two successive T-mazes with return corridors
- Decision points T1 and T2
- Rewarding food pellets at each side



Experimental setup

[Aubin et al., 2018] set up an experimental task environment.

- Two successive T-mazes with return corridors
- Decision points T1 and T2
- Rewarding food pellets at each side



- Three tasks were given:
  1. go to left side
  2. go to right side
  3. alternate between left and right

Experimental setup

The agent's state is a concatenation of 32 location components and 2 reward memory components (left and right).

A problem was encountered, namely that some states may have more than one predecessor.

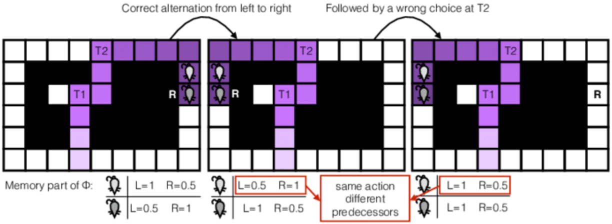
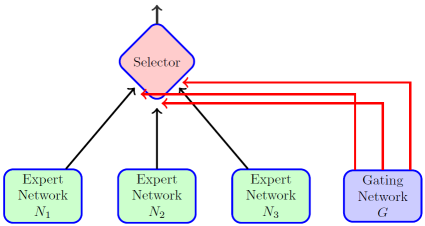


Figure: Multiple predecessors for same location[Aubin et al., 2018]

GALMO

To confront this issue, a *growing algorithm to learn multiple outputs* (GALMO) was created.

- Algortihm creates multiple neural neworks N_i coupled with a gating network (*mixture of expert* [Jacobs et al., 1991])
- Creates a new network on the fly when outlier is detected (Assumption: outlier is caused by an input that should predict multiple outputs). Specializes each network to a predecessor



Dyna-Q

Second part of the algorithm was neural *Dyna-Q* with *prioritized sweeping*.
Dyna is an architecture that mixes direct RL updates and planning updates:

- Agent creates a model from environment experience during initial exploration
- Policy is updated using simulated experience from the model and actual experience

Dyna-Q

Second part of the algorithm was neural *Dyna-Q* with *prioritized sweeping*.

Dyna is an architecture that mixes direct RL updates and planning updates:

- Agent creates a model from environment experience during initial exploration
- Policy is updated using simulated experience from the model and actual experience

  $\Longrightarrow$ Simulated experience replays are treated as model for hippocampal replays!

Outline

GALMO Results

A framework to analyze the role of hippocampal replays in the learning process was desired.

The state activation analysis has shown that around 15-20% of state reactivations during replays correspoded to actual 3-step sequences.
Unfortunately, no clear pattern could be found comparing the state reactivations during the three tasks.

GALMO Results

The network architecture proposed by [Aubin et al., 2018] was able to learn multiple predecessor cases. Overall, the system solved all tasks faster than the corresponding Q-learning system.

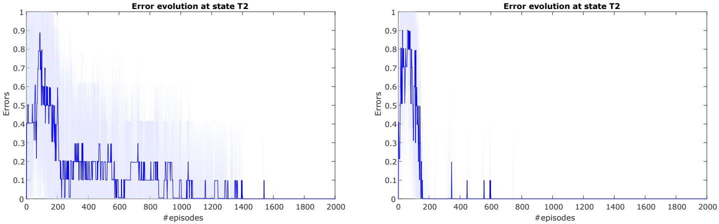


Figure: Learning without and with replays [Aubin et al., 2018]

Unfortunately, no information was given on the duration of the networks' learning, which would be interesting concerning a cost-benefit calculation.

Project

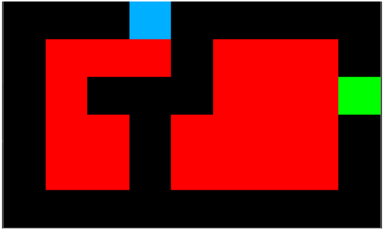
As a practical part, the experimental environment and Q-learning were implemented in Python. The goal was

- to enhance the understanding of structures and procedures in Reinforcement Learning
- to create comparative results of a Q-learning implementation

Project

As a practical part, the experimental environment and Q-learning were implemented in Python. The goal was

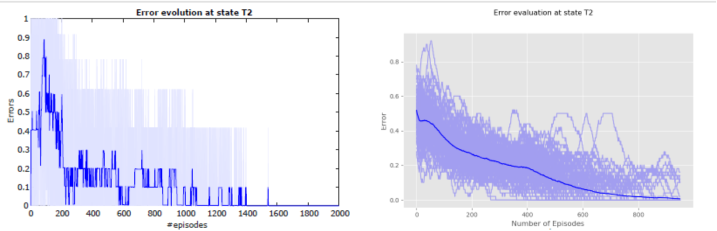
- to enhance the understanding of structures and procedures in Reinforcement Learning
- to create comparative results of a Q-learning implementation



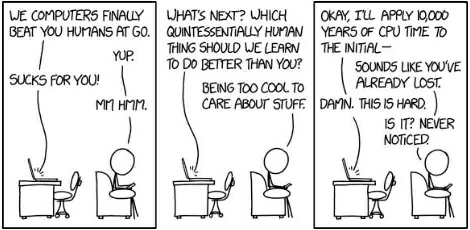
Project

As a practical part, the experimental environment and Q-learning were implemented in Python. The goal was

- to enhance the understanding of structures and procedures in Reinforcement Learning
- to create comparative results of a Q-learning implementation



The own results were slightly better than the corresponding Q-learning due to initial modification, but ultimately can not surpass the Neural Dyna-Q with prioritized sweeping.



Thank you for your attention!

Aubin, L., Khamassi, M., and Girard, B. (2018).
Prioritized sweeping neural dynaq with multiple predecessors, and hippocampal replays.
In *Living Machines 2018*, LNAI, page TBA, Paris, France.

Dishpande, A. (2016).
Categories of machine learning.

Hatem, M. and Abdessemed, F. (2009).
Simulation of the navigation of a mobile robot by the qlearning using artificial neuron networks.
volume 547.

Irpan, A. (2018).
Deep reinforcement learning doesn't work yet.
`https://www.alexirpan.com/2018/02/14/rl-hard.html`.

Jacobs, R., Jordan, M., Nowlan, S., and Hinton, G. (1991).
Adaptive mixture of local expert.
Neural Computation, 3:78–88.

Maguire, E., Nannery, R., and Spiers, H. (2006).