

Final project Reinforcement Learning

Master's degree in Artificial Intelligence and Robotics

Course: Reinforcement Learning

Professor: Roberto Capobianco

Authors:

Alessandro Lambertini – 1938390

Denise Landini – 1938388

La Sapienza University - Roma

Academic year: 2020-2021



SAPIENZA
UNIVERSITÀ DI ROMA

Summary

1. DESCRIPTION OF THE METHOD AND IDEA OF THE PAPER.....	3
2. IMPLMENTATION OF THE PAPER PROJECT	3
2.1. METHODS USED.....	3
2.1.1. Q-LEARNING.....	3
2.1.2. DEEP Q-LEARNING.....	3
2.2. ENVIRONMENT AND PARAMETERS USED FOR REIMPLEMENTATION	4
2.2.1. CART-POLE.....	4
2.2.2. TAXI.....	5
2.3. LIMITATION AND STRENGTHS OF THIS METHOD	5
2.3.1. LIMITATION.....	5
2.3.2. STRENGTHS	5
3. RESEARCH PART	6
3.1. BUILD COMPLEX ENVIRONMENT	6
3.2. PROBLEM IN THE PAPER WITH OUR COMPLEX ENVIRONMENT.....	6
3.3. FIRST EXPERIMENT.....	6
3.4. SECOND EXPERIMENT	7
3.5. THIRD EXPERIMENT.....	8
3.6. OTHER IMPLEMENTATIONS.....	8

1. DESCRIPTION OF THE METHOD AND IDEA OF THE PAPER

The idea of the paper to address the problem of explaining the behaviour of a machine learning algorithm is by asking to the agent what chain of events intend to happen as a result of a particular action choice.

Instead of predicting an agent's future state by rerunning repeated forward simulations from the same initial state, the paper proposes to sum the past events weighted by the importance that the agent put on them while learning. So, the agent's intention is based on the behaviour policy.

To know why an RL agent prefers one action instead another, we need to infer the agent's implicit estimate of $p(s_{t+n}|s_t, a_t, \pi)$ with $n > 0$. With this, we wish to know the expected feature state that leads an agent to decide that one action in the current state would be preferable to another without understanding the internal computation.

The approach followed by the paper augment standard RL learning method by adding a simple map of discounted expected state H , called belief map, that needs to be learned during the training phase and preserve additional information. We also need to choose an update rule for H that match the update equation of the learning method to guarantee consistency between the Q values and the expected feature state.

2. IMPLMENTATION OF THE PAPER PROJECT

2.1. METHODS USED

In order to do the reimplementaion we have used two methods:

- Q-Learning;
- Deep Q-Learning.

2.1.1. Q-LEARNING

For Q-learning we update H with the given update rule:

$$H(s_t, a_t) \leftarrow H(s_t, a_t) + \alpha \left(1_{s_t, a_t} + \gamma H(s_{t+1}, \arg \max_{a \in A} Q(s_{t+1}, a)) - H(s_t, a_t) \right)$$

where:

$H(s_t, a_t) \in \mathbb{R}^{S \times A}$ is the expected discounted sum of feature state visitation.

α is the learning rate.

$1_{s_t, a_t} \in \mathbb{R}^{S \times A}$ denotes a binary indicator function which has 1 at position s_t, a_t and 0 elsewhere.

γ is the discount factor.

Q-learning formula used:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t) \right)$$

where r is the reward.

So, every time we update the Q -value, we also update H (belief map).

2.1.2. DEEP Q-LEARNING

In Deep Q-Learning we do not use a belief map, but we have a Deep Belief Network trained using the same hyperparameters as DQN.

Layer	Type	Input	Size
input	N/A	N/A	4
fc1	MLP	input	128
fc2	MLP	fc1	512
output	MLP	fc2	2

Layer	Type	Input	Size
input	N/A	N/A	162
fc1	MLP	input	500
fc2	MLP	fc1	2000
output	MLP	fc2	2, 162, 2

Figure 1: Size of DQN and DBN for Cart-Pole

Layer	Type	Input	Size
input	N/A	N/A	4
fc1	MLP	input	500
fc2	MLP	fc1	2000
output	MLP	fc2	6

Layer	Type	Input	Size
input_belief	N/A	N/A	500
input_action	N/A	N/A	6
belief_stream	MLP	input_belief	1024
action_stream	MLP	input_action	128
concat	N/A	input_belief, input_action	1024+128 = 1152
fc3	MLP	concat	2048
output	MLP	fc3	500

Figure 2: Size of DQN and DBN for Taxi

2.2. ENVIRONMENT AND PARAMETERS USED FOR REIMPLEMENTATION

We have reimplemented the paper in two environments using Q-learning and DQN (Deep Q-Network):

2.2.1. CART-POLE

It is a continuous control problem that is characterized by car position, car velocity, pole angle and pole velocity. It has two possible actions: left and right.

We discretize the state space in bins of (3, 3, 6, 3).

In particular, in this environment we have a Pole attached by an unactuated joint to a Cart.

The pendulum starts upright, and the goal is to prevent it from falling over.

	Q-learning	DQN
Learning rate α	$\alpha = 0.1$	$\alpha = 0.0001$
Discount factor γ	$\gamma = 1$	$\gamma = 1$
Initial exploration probability ϵ (linearly decreased throughout 500 episodes)	$\epsilon = 1$	$\epsilon = 1$
Final exploration probability ϵ	$\epsilon = 0.1$	$\epsilon = 0.05$
Optimizer	/	Adam optimizer with gradient clipping at $[-1, 1]$ Batch 16 experiences together for optimization.
Loss	/	Huber loss with discount factor $\gamma = 1$

The episode terminates:

- when the length of the episode reaches 200 times step;
- when the Pole falls.

In this case, we can observe from the belief map that only the state where the car is in the middle are chosen.

2.2.1.1. RESULTS OBTAINED

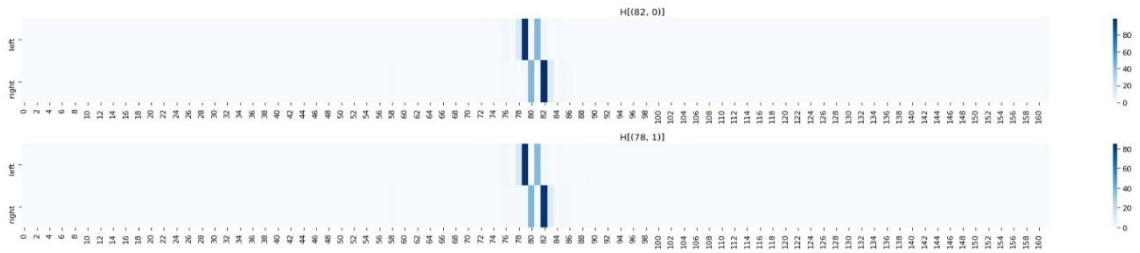


Figure 3: Belief Map for Cart-Pole simulation in two different states with action left and right. In the plots we can see that the Cart will remain in the central area of the environment. We can understand this because we have high value in the central part and zero value elsewhere.

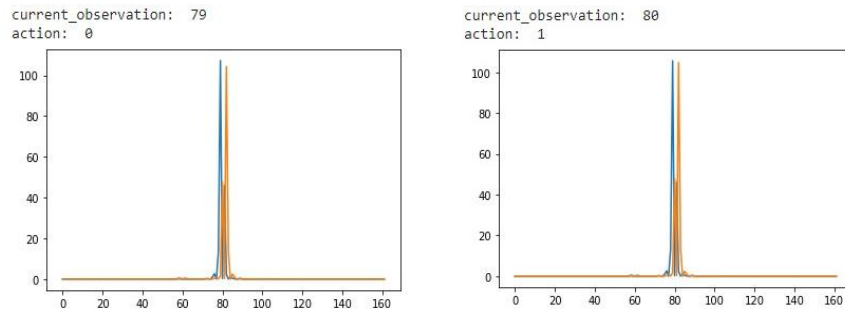


Figure 4: Different visualization of belief map of Cart-Pole

2.2.2. TAXI

It starts from a random initial position, and it navigates around the obstacles in order to collect a passenger from a random location and navigate to a random destination.

	Q-learning	DQN
Learning rate α	$\alpha = 0.4$	$\alpha = 0.0001$
Discount factor γ	$\gamma = 0.9$	$\gamma = 0.9$
Exploration probability ϵ (linearly decreased throughout 250 episodes)	$\epsilon = 1$	$\epsilon = 1$
Final exploration probability ϵ	$\epsilon = 0.1$	$\epsilon = 0.1$
Optimizer	/	Adam optimizer
Loss	/	Huber loss with discount factor $\gamma = 1$

The episode terminates:

- when the length of the episode reaches 200 times step;
- when the goal is reached.

2.2.2.1. RESULTS OBTAINED

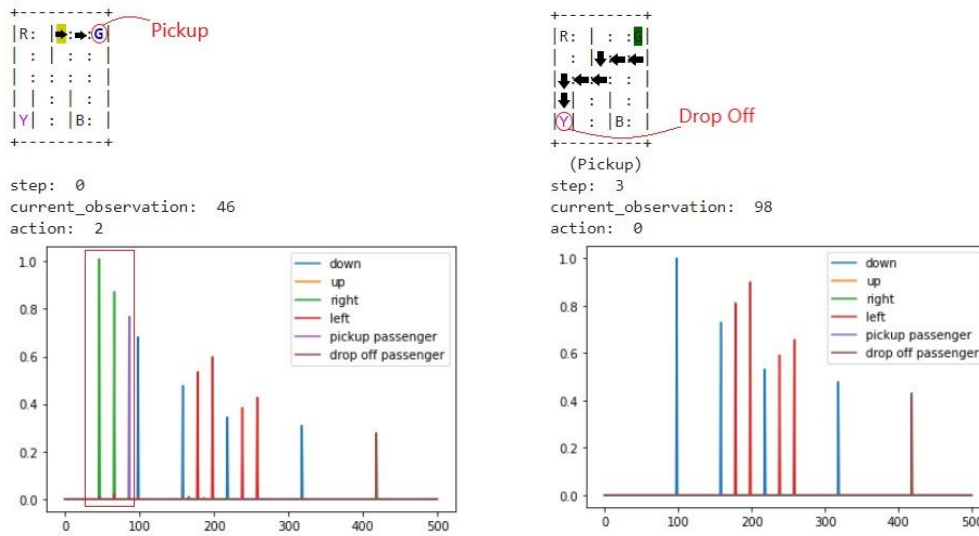


Figure 5: plots of the belief map of taxi with environment representation. Higher is the value of the action and higher is the confidence that the agent will visit that state. In the environment plot there are also drawn the action performed and the state visited based on the belief map.

In the left plot we have the first three actions, in the right we have the remaining actions.

2.3. LIMITATION AND STRENGTHS OF THIS METHOD

2.3.1. LIMITATION

This approach allows to explain a single instance of an agent policy, but this cannot be applied in a more general case. So, every time we train an agent, we need to learn another time the belief map H .

Because how is build the update rule for H , it works well only when tabular Q-learning works well. So, it is best suited for low-dimensional and easily visualizable problem. We need small environment also because it uses a lot of memory.

2.3.2. STRENGTHS

It can work in every environment in which tabular Q-learning works well and it is easily implementable.

3. RESEARCH PART

3.1. BUILD COMPLEX ENVIRONMENT

We started by creating a more complex environment to check the limit of the belief map. The environment is a big maze with moving enemies.

At each epoch, the agent starts from a random position and must reach the goal without being eaten by an enemy.

The agent has 12 different actions:

- vertical and horizontal movements (up, down, left, right);
- diagonal movements (top-right, top-left, ...);
- agent can shoot in vertical and horizontal directions.

The agent gets a bad reward every time it hits a wall, a map limit, miss the shoot or if it is eaten by an enemy.

The goal of the agent is only to reach the end of the maze.

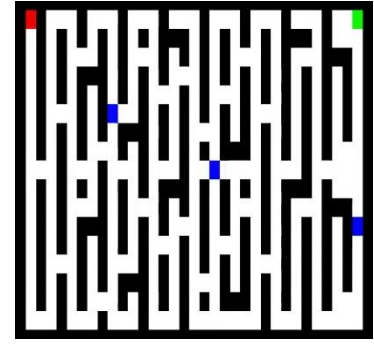


Figure 6: Complex environment.

Green point: goal
Blue points: enemies
Red point: agent

3.2. PROBLEM IN THE PAPER WITH OUR COMPLEX ENVIRONMENT

The problem that we have encountered for the belief map is that after shooting to one enemy, the belief map of this state and the successive states are full of not relevant information.

This is due to:

- how is made the environment;
- how is updated the belief map.

In fact, when the agent shoots to the enemy, and since we are in the training phase, the agent explore all the areas. All these areas that he explores can only be visited after that the enemy has been killed. So, when he finds the optimal path to reach the goal state, these areas will no longer be updated.

However, this is not the case when we have not yet killed an enemy, because in this case the agent can start an epoch in any position. So, all the areas that the agent can visit will be updated.

In conclusion, since after killing the enemy there are areas that we visit a few times and then we can no longer revisit them, we have this information that is superfluous in the belief map¹.

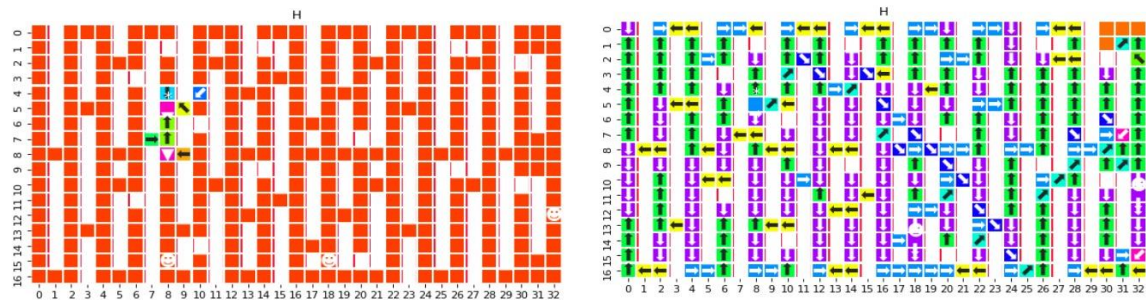














Figure 7: in the first image the agent is in the position (5,8) and he is going to shoot bottom to the enemy that is in position (15,8). In the second image, the agent is in position (5,8) and it has just killed an enemy. How we can see, the belief map in the right plot is full of irrelevant information because of the reason explained before.

3.3.FIRST EXPERIMENT

First, we try to group up the actions in three sets:

- set 1: vertical and horizontal movements (up, down, left, right);
- set 2: diagonal movements (top-right, top-left, ...);
- set 3: agent can shoot in vertical and horizontal directions.

¹ Legend: [left: , top-left: , top: , top-right: , right: , bottom-right: , bottom: , bottom-left: , shoot-left: , shoot-top: , shoot-right: , shoot-bottom: 

We use the same update rule.

Even with these changes, we encountered the same problem that we had in the first case and the belief map that we get gives us more possible series of actions that the agent could do, but we have less precision about what it will actually do.

We also tried to group all the actions into one to see what happened to the belief map², but the result we got was not useful.

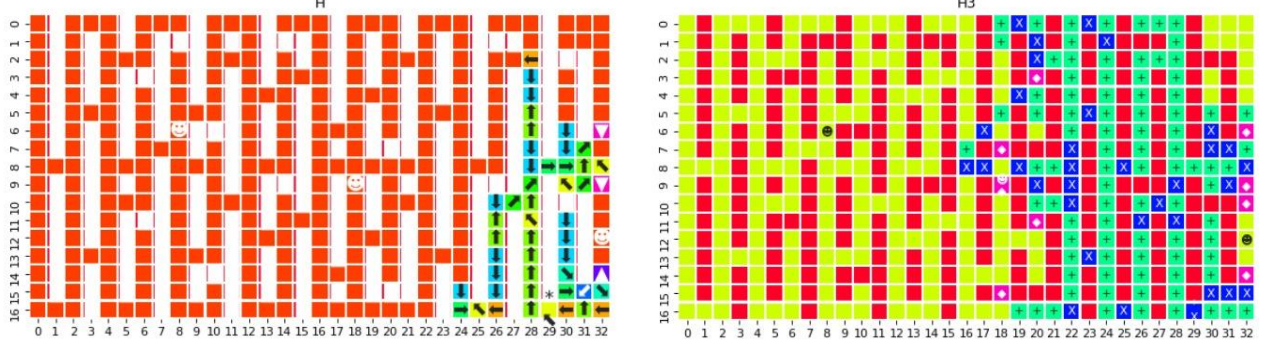


Figure 2: As we can see looking at the right image, we have a lot of different paths to follow starting from the position of the agent. So, the belief map gives us more possibility but less accuracy to what the agent could achieve.

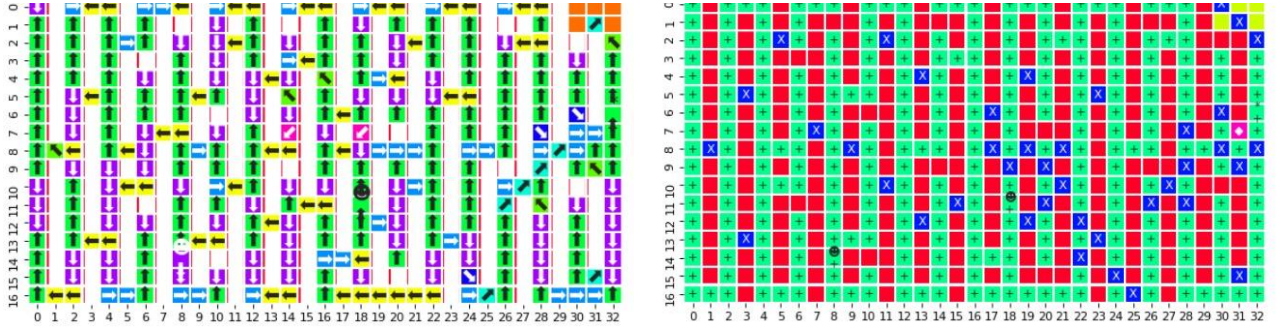


Figure 3: figure shows that the problem we had in the belief map with the std update rule is also present in this case.

3.4. SECOND EXPERIMENT

We have tried to change the update rule in this way:

$$H(s_t, a_t) = k_d \left(1_{s_t a_t} + H(s_t, a_t) + 1_{s_t a_t} H(s_{t+1}, \arg \max_{a \in A} Q(s_{t+1}, a)) \right) + k_p e(s_t, a_t)$$

$$e(s_t, a_t) \leftarrow \frac{e(s_t, a_t)}{\mu(H(s_t, a_t))} \left(H(s_t, a_t) - H(s_{t+1}, \arg \max_{a \in A} Q(s_{t+1}, a)) \right)$$

where:

k_d, k_p are two factors that regulates the importance between the current update and the error. They should be chosen small and positive.

For the error $e(s_t, a_t)$, we choose to update the error with the difference between the belief map of this state and the future state, normalized by the mean μ .

For this formula, we are inspired by the control law of robotics to obtain a linear damping of the error, but we have obtained a worst result with respect to which of the paper.

² Legend: [vertical and horizontal movements: +, diagonal movements: X, shoot in vertical and horizontal directions: ♦]

3.5. THIRD EXPERIMENT

Inspired by the second experiment, we have decided to consider the value of the expected reward to cut off the information from the far and not significant path. So, we have inserted it in the update rule that we have already used for the belief map, and we came up with this new update rule:

$$H(s_t, a_t) \leftarrow H(s_t, a_t) + \alpha \left(1_{s_t, a_t} + \gamma H(s_{t+1}, \arg \max_{a \in A} Q(s_{t+1}, a)) - H(s_t, a_t) \right) + k_q \left(r + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t) \right)$$

where k_q is a small number that means how much we consider the expected reward.

We choose to use $k_q = 0.01$ because we want that the expected reward does not affect so much our belief map but only cut off the not relevant information.

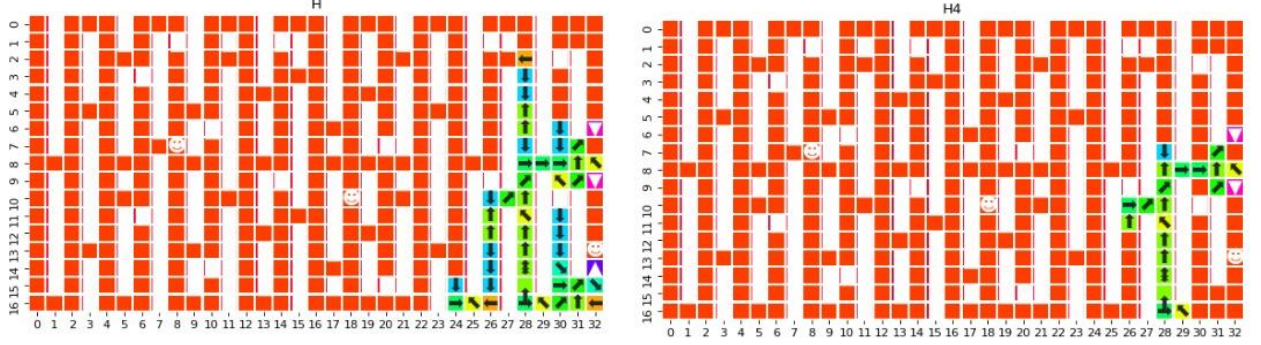


Figure 10: we can see from the two plots that in the second case we can better see what the agent is trying to achieve (more precision).

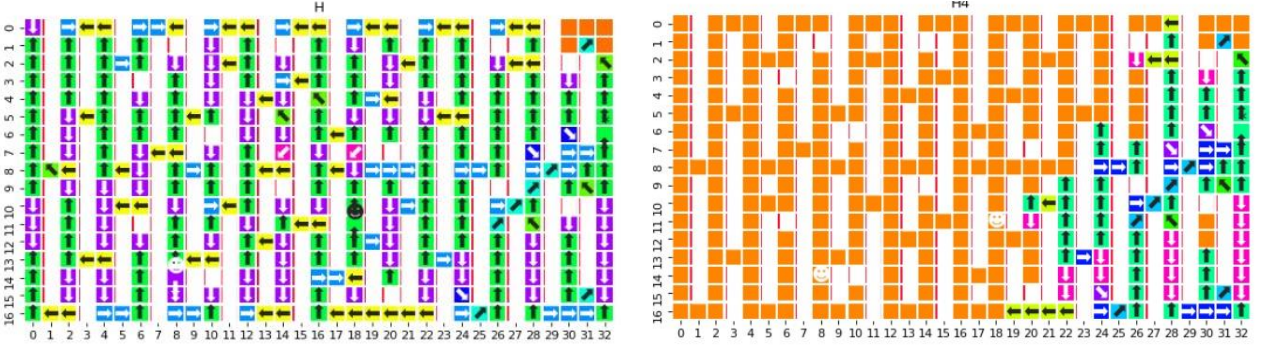


Figure 11: this is the plot of the belief map in the state in which we have the problem, so when the agent has just killed an enemy. We can see that our update rule gives a better result with respect to the paper's update rule.

3.6. OTHER IMPLEMENTATIONS

We also test our update rule in Cart-Pole and Taxi to check if it works well in other environment and we obtain similar result.

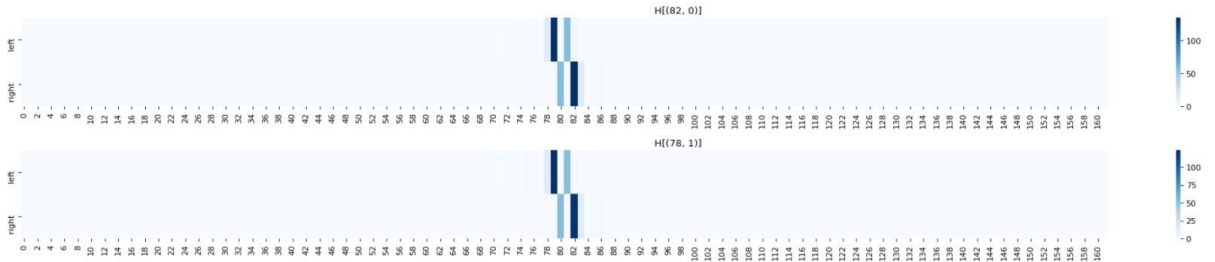


Figure 3: plot of the belief map of our update rule. If we compare this figure with figure 3, we can see that the result is similar.

We also implement the belief map for Cart-Pole and Taxi using as update rule SARSA and not Q-learning. In order to do this, we set the update rule as:

$$H(s_t, a_t) \leftarrow H(s_t, a_t) + \alpha (1_{s_t, a_t} + \gamma H(s_{t+1}, \pi(s_t)) - H(s_t, a_t))$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha (r + \gamma Q(s_{t+1}, \pi(s_t)) - Q(s_t, a_t))$$

We change H in this way in order to maintain consistency between H and Q.

The result obtained for belief map, using SARSA, are like those obtain with Q-learning. So, we can conclude that this approach can be used also with other algorithms.

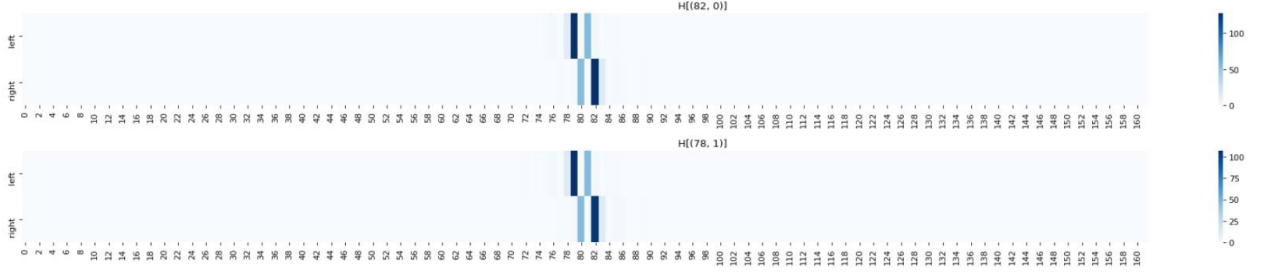


Figure 13: plot of the belief map using SARSA.

If we compare this figure with figure 3, we can see that the result is similar.