Half Field Offense

In this experiment I introduced more actions and more features. Now the agent has 5 possible actions, and one extra feature, which tells to the agent where is the ball at. Also, I changed some things in the learning process.

Why use more actions?

Before I was using **only three very high level actions** (Move, Dribble and Shoot). These were already “intelligent” actions, since they decided the direction and angles, by using metrics. Which ended up generating quite good players, although they would not explore the space properly. So, in order to create a real Q-learning agent, I started to use 5 discrete actions (Move up, Move left, Move down, Move right, and Kick to goal).

What changed in the learning process?

Since there are only sparse rewards, in the end of the game the agent would receive a positive or negative reward. So, when updating the q-learning table, the last action would be the most affected one by the reward. Although, if, for example, the agent shoots and scores a goal, the last action is not the Shoot one. After shooting the agent will keep moving and doing other stuff, which do not contribute to score the goal.

So I decided to only update the q-learning table from the moment that the agent first has the ball, until he shoots for the last time. Also, in the end of the game I update the q-learning table on the inverse order to more quickly propagate the reward.

Bugs to investigate further

There are some strange behaviors, specially when kicking the ball. Sometimes, the agent shoots the ball in random directions, despite I specify to which direction should the agent shoot. I read the manual, and I could not find nothing about it. I have to debug it in the future.

1. Game parameters:

- Number of **teammates: 0;**
- Number of **opponents: 1;** (Dumb Goalie)
- Number of **episodes: 10k train;**

2. Q-Learning Agent

2.1. Q learning parameters:

- Learning rate: 0.10;
- Epsilon values: [0.8, 0.7, 0.5, 0.4, 0.3];
- Discount factor: 0.99;
- **Q learning table dim:** 120 environment states * **5** number of actions;

2.2. State Features

1. **Position** – int [0, 6], subdivided the field in 6 regions, each integer signalizes one of these regions;
2. **Should Shoot** – int (0,1). Agent's goal opening angle > 20%;
3. **Opponent is close** – int (0,1). Opponent is close;
4. **Ball Position –** int [0, 4]. Five states {0: "Player Has Ball", 1: "Up", 2: "Right", 3: "Down", 4: "Left"}

2.3. Actions

1. **Dribble/Move Up –** The agents goes up. If it has the ball, it dribbles the ball up, otherwise just moves up.
2. **Dribble/Move Down** – The agents goes up. If it has the ball, it dribbles the ball up, otherwise just moves up.
3. **Dribble/Move Right** – The agents goes up. If it has the ball, it dribbles the ball up, otherwise just moves up.
4. **Dribble/Move Left** – The agents goes up. If it has the ball, it dribbles the ball up, otherwise just moves up.
5. **Kick to Goal –** Shoot to the goal. This command only works when the agent has the ball.

2.4. Rewards

- - 1 – the game ends without scoring goal;
- + 1 – score goal;

3. Goalkeeper Agent

The agent presents 2 simple behavior:

If ball is on the upper side of the field:

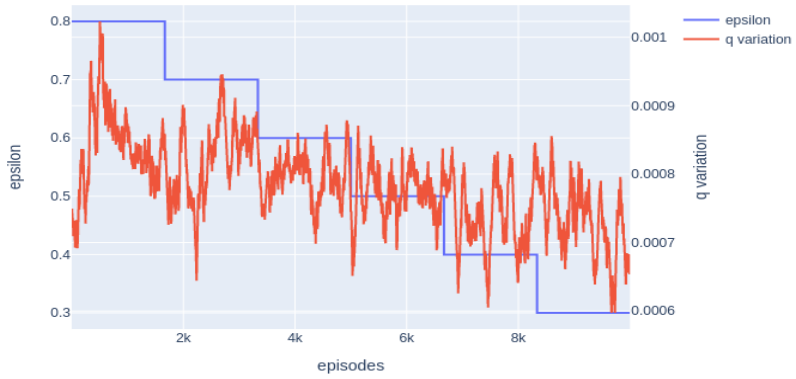
the goalkeeper moves to the goal top corner;

Else:

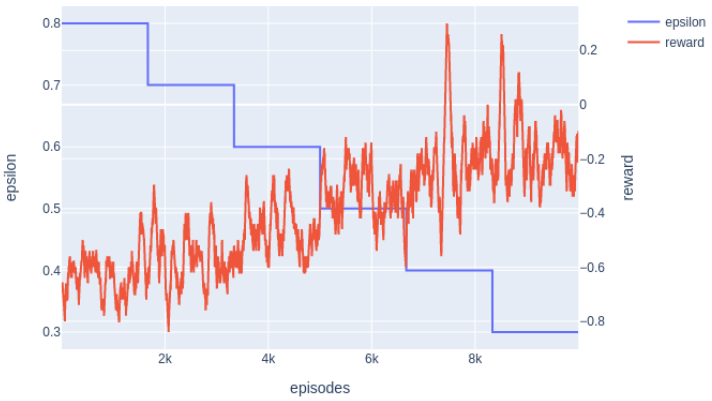
the goalkeeper moves to the goal bottom corner;

4. Results:

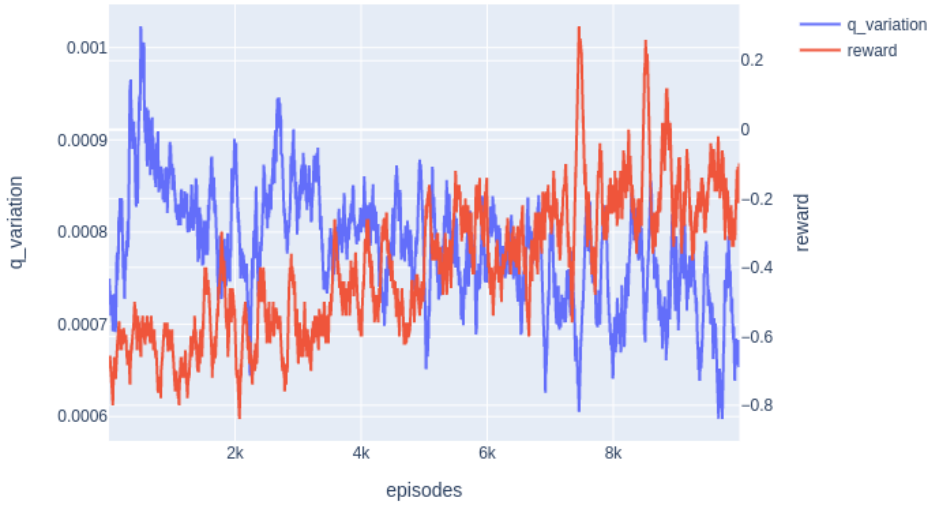
4.1. Relation q-learning variation – epsilon variation:



4.2. Relation epsilon variation – reward variation:



4.3. Relation q-table variation – reward variation:



4.3 Exploration Heat-map:

On the y-axis is the 6 different regions, on the x-axis bar are the 5 different actions. This sums the total actions selected at each state, during all the training.