COMP3611: Reinforcement Learning

MACHINE LEARNING

Ilyass Taouil [sc14it]

University of Leeds

Domain

Figure 1: Domain of the game.

The domain consists of 24 states, precisely 4 rows and 6 columns. The game has various elements to it, these are listed below with their respective value functions:

- **Ice cream**
  Positive reward of +10

- **Treasure**
  It represents the goal with a positive reward of +100 (absorbing state)

- **Fire**
  Negative reward of -20

- **Pit**
  Negative reward of -100 (absorbing state)

- **Puddle**
  Negative reward of -5

- **Friends**
  Positive reward of +10

Experiments

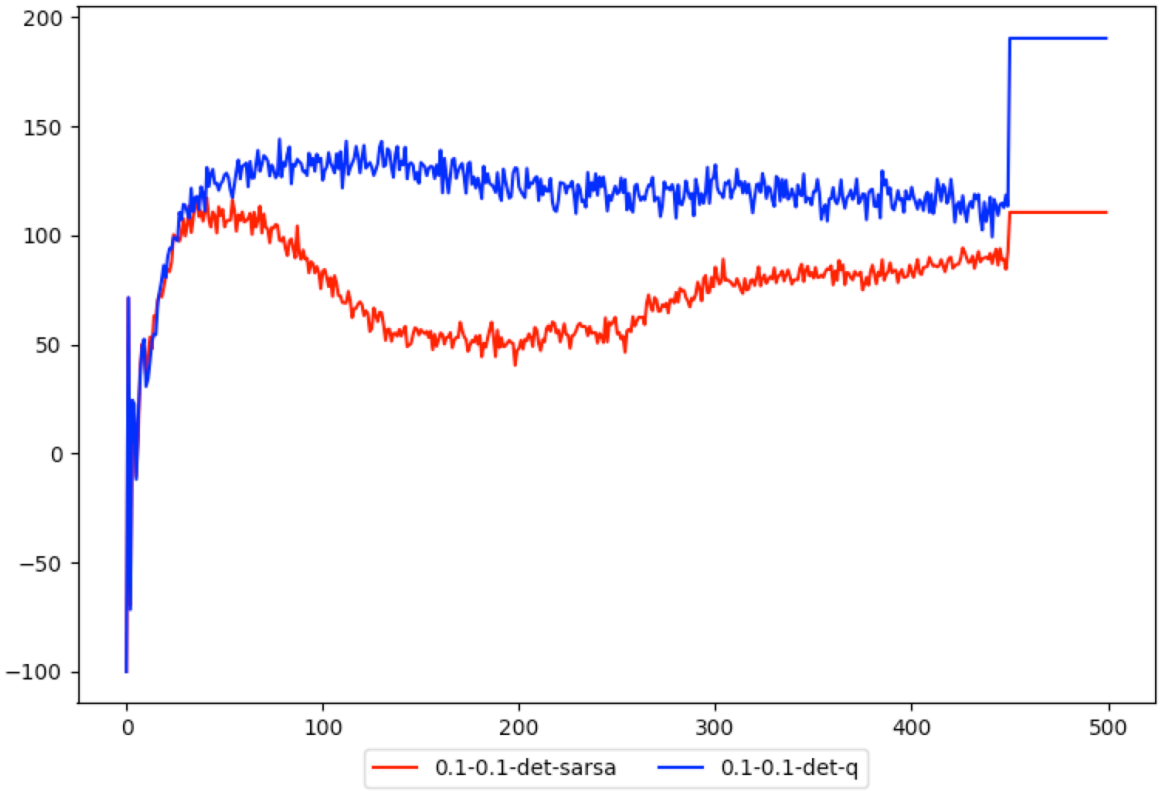

Figure1: 0.1-0.1 deterministic SARSA & QLEARNING

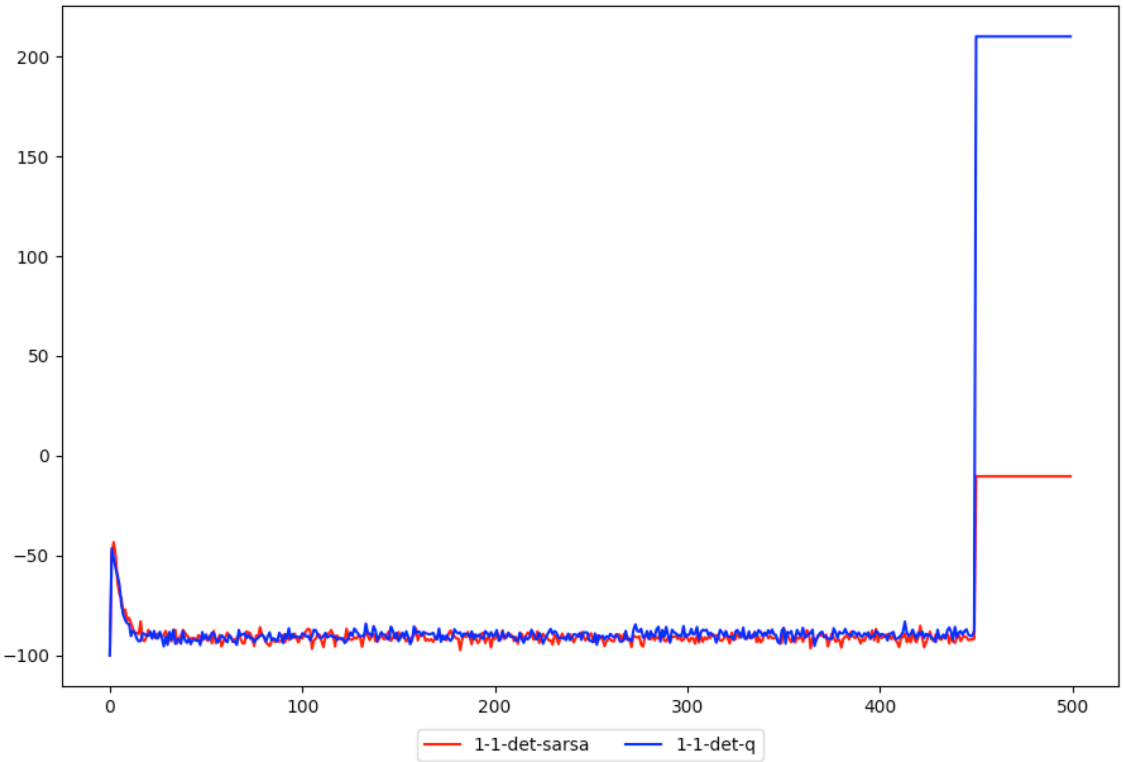

Figure2: 1-1 deterministic SARSA & QLEARNING

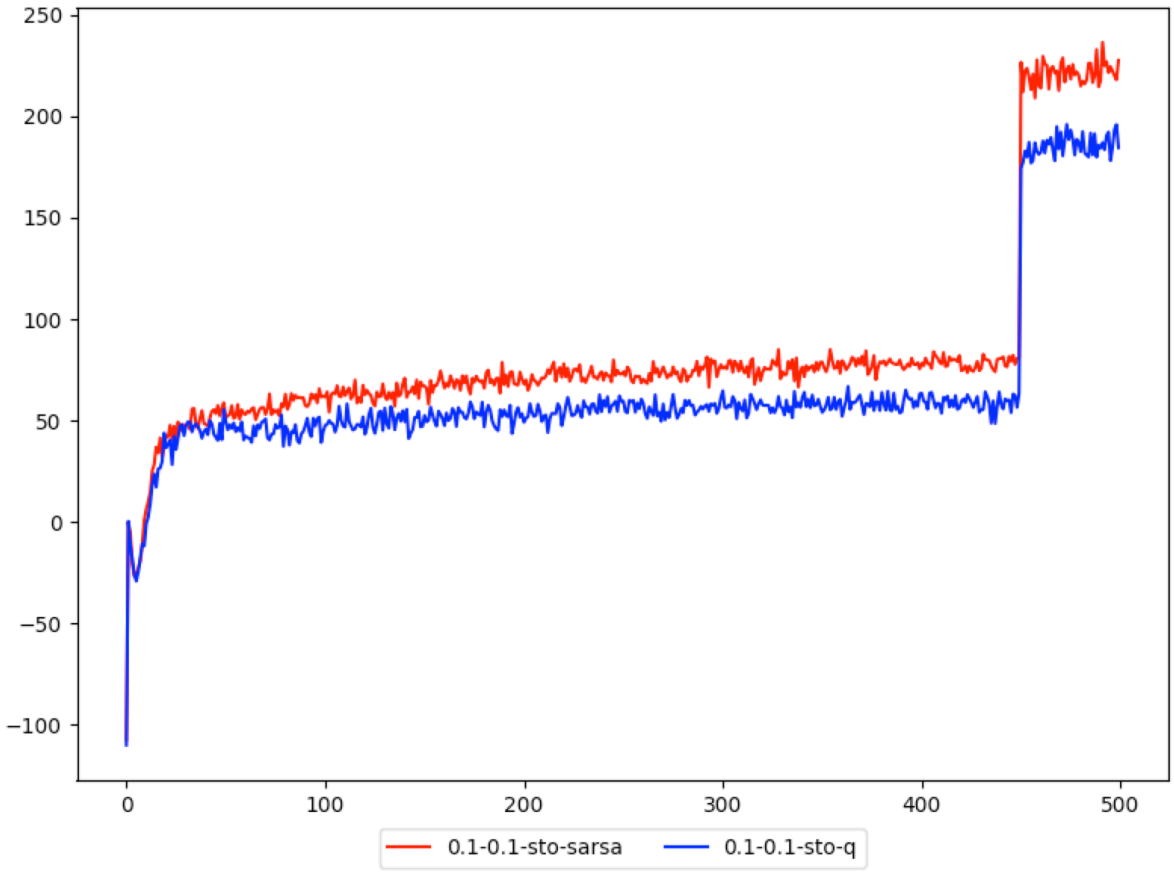

Figure3: 0.1-0.1 stochastic SARSA & QLEARNING

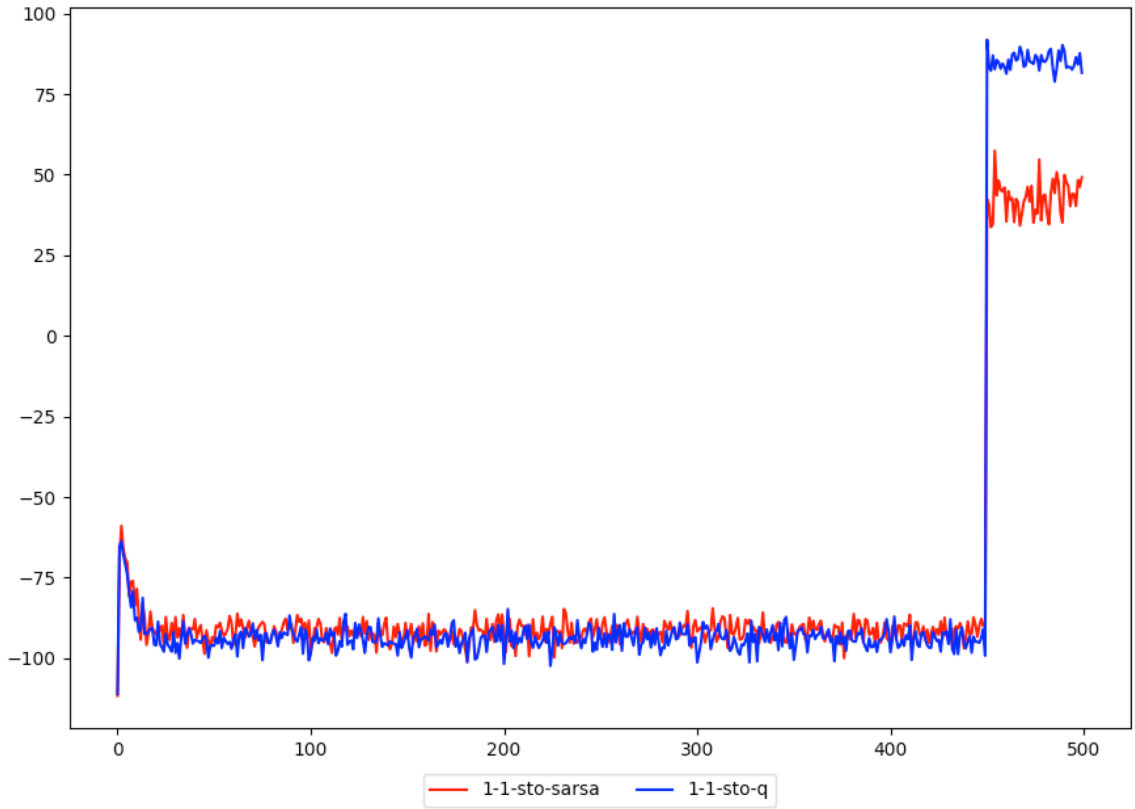

Figure4: 1-1 stochastic SARSA & QLEARNING

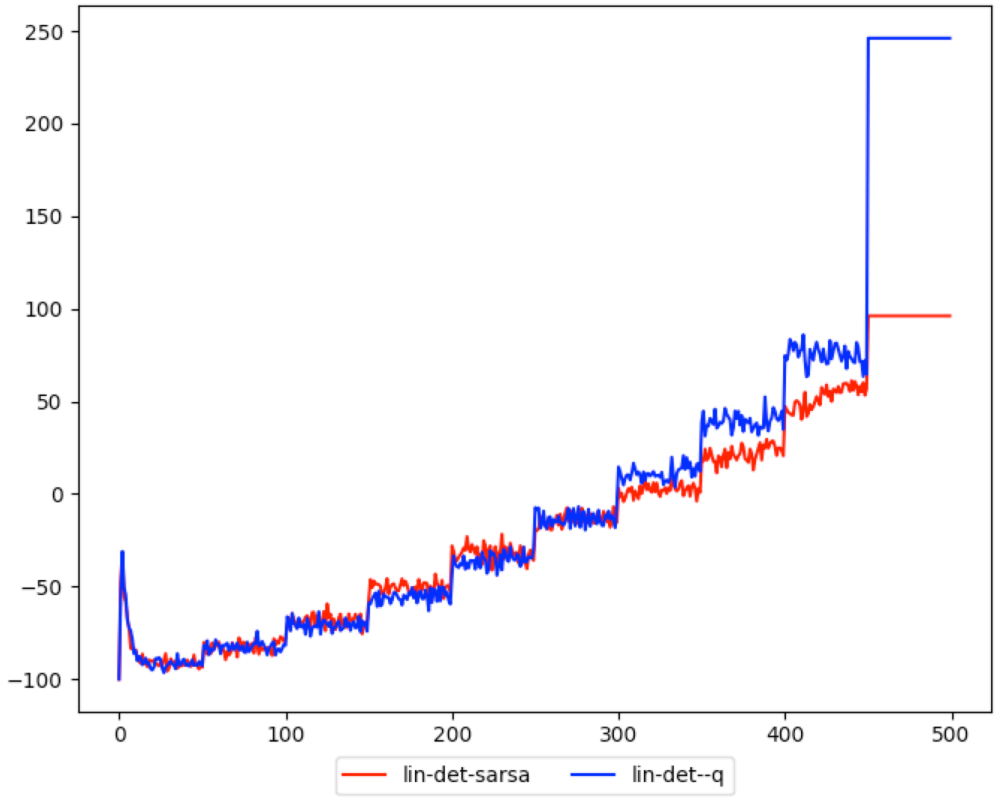

Figure5: linear deterministic SARSA & QLEARNING

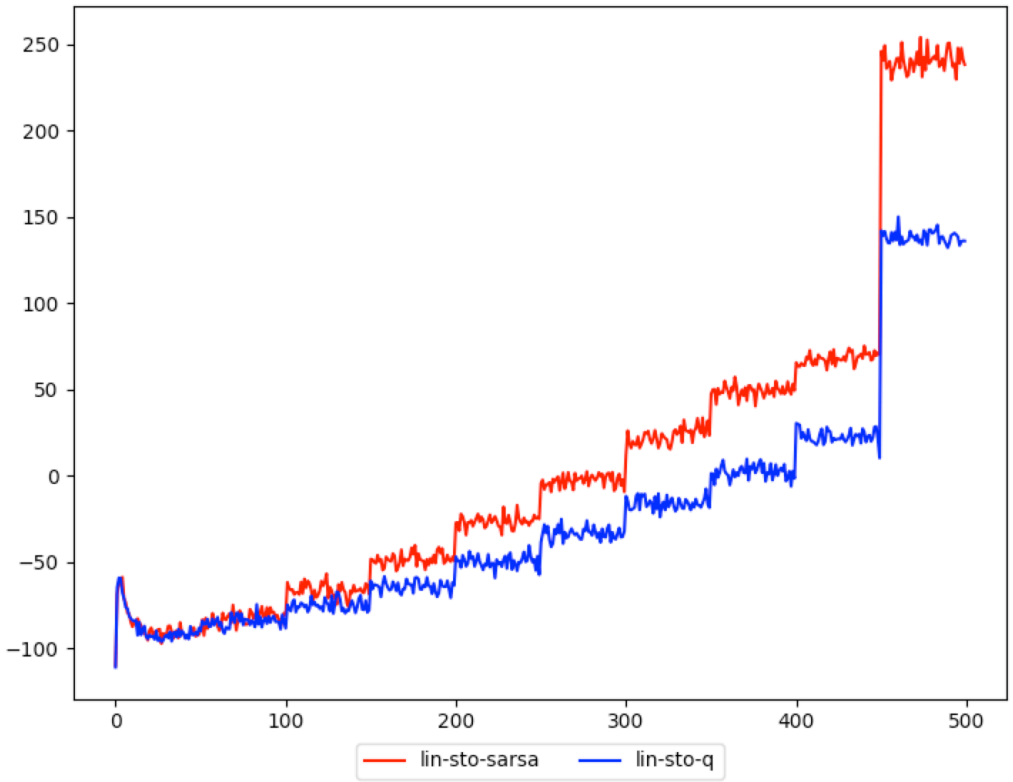

Figure6: linear stochastic SARSA & QLEARNING

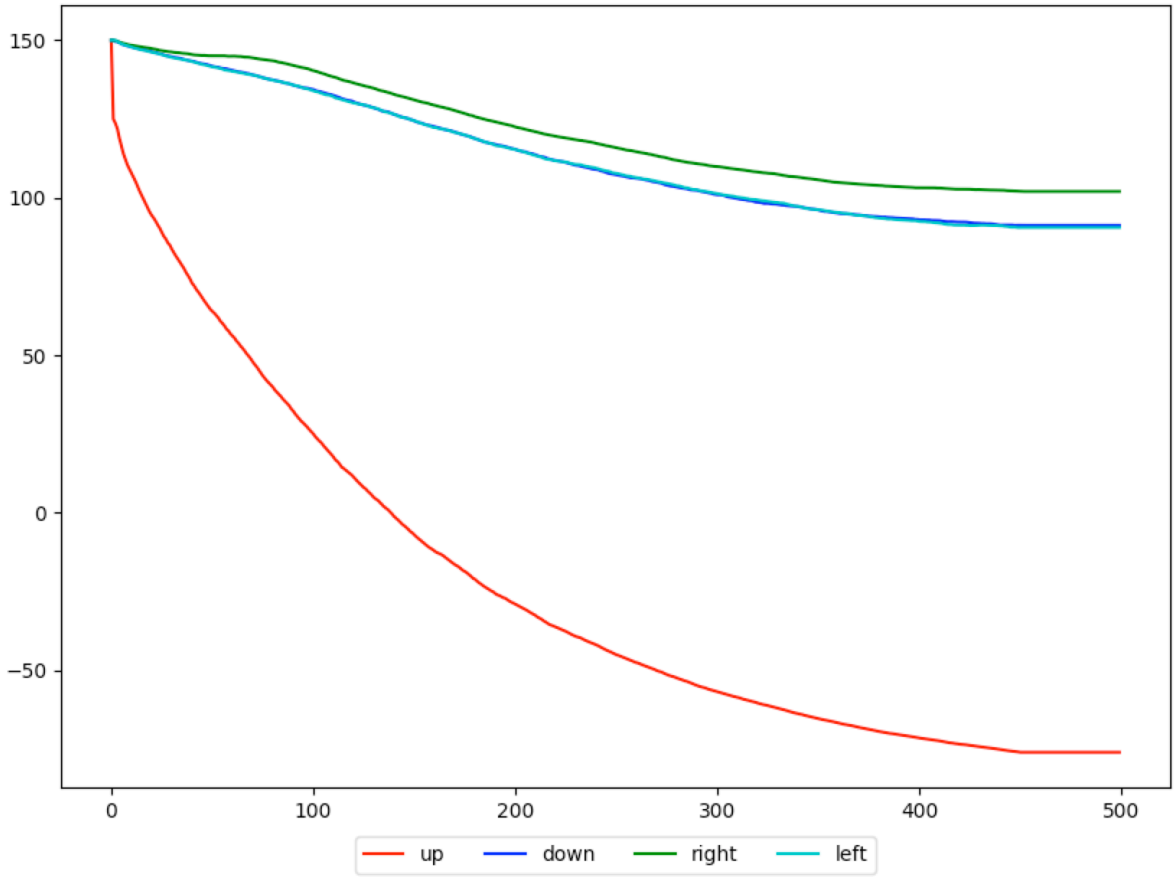

Figure7: action-value function for 0.1-0.1 deterministic for SARSA & QLEARNING

- **Policy quality**

  The policy learned in the various experiments run show a different result depending on the algorithm type used and the hyper parameters values.

  In Figure1 both algorithms converge to a solution within the set 500 EPOCHS, with SARSA being able at the end of the iterations to get to the goal getting the maximum reward available with the least amount of steps, while QLEARNING still gets to the goal but using the maximum number of steps available before ending in the goal, resulting in a higher number of rewards.

  In Figure2, we can notice how by increasing the alpha and epsilon parameter the SARSA algorithm fails to lean an optimal policy with its highest rewards being around -20, while QLEARNING is still able to learn an optimal policy. This behaviour is a direct consequence of the increase of the two hyper parameters that influence the learning process and the on-policy behaviour of SARSA.

In Figure3 the stochastic behaviour does not affect the two algorithms performance as they still converge, nonetheless we can notice how after the two converge there is still a bit of noise and this depends is because of the stochastic behaviour of the movement choice.

In Figure4 the stochastic behaviour improves upon the deterministic one for the same set of parameters show in Figure2. In fact, both algorithm now learn a better policy, although not the optimal one.

In Figure5 and Figure6, both learning algorithm's reward grow linearly to a rather good policy. However, we can notice how QLEARNING and its **direct** learning reaches a good policy for both stochastic and deterministic movement while SARSA does not learn an as good policy as QLEARNING deterministic approach

In Figure7 we notice how the QVAL values respect the correct update of it as the agent learns. In fact, a movement up would make the agent fall into the pit, while going down or left would just make the agent get a 0 reward, hence the similar curve, while the right action makes the agent reach the goal and in fact it holds the highest value.

- **Alpha** & **Epsilon**

  The alpha parameter represents the learning rate, which can have the positive effect of speeding the learning process or making it never to converge to an optimal policy which for SARSA that follows an on-policy behaviour can signify to never learn an optimal one. QLEARNING, however is not as sensitive to the high values of alpha as SARSA is for its off-policy behaviour.

  The epsilon parameter determines the probability of taking a random action for the exploration process, the higher it is the more probable. High values of epsilon can put off the SARSA on-policy behaviour because of the high randomicity of the agent choices.