REINFORCEMENT LEARNING

deboshre - 50291550

December 5, 2018

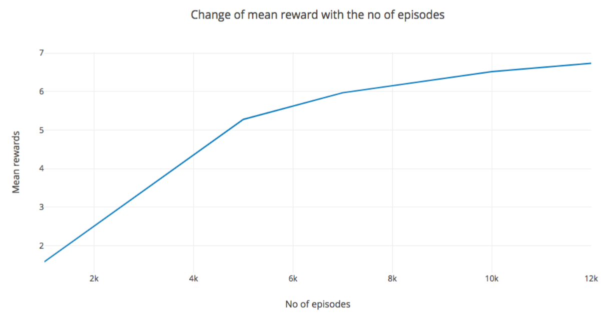
1 Introduction

The project deals with the application of reinforcement learning through Deep-Q-Network method.The environment has a grid system where Tom(agent) is trying to catch Jerry(goal).

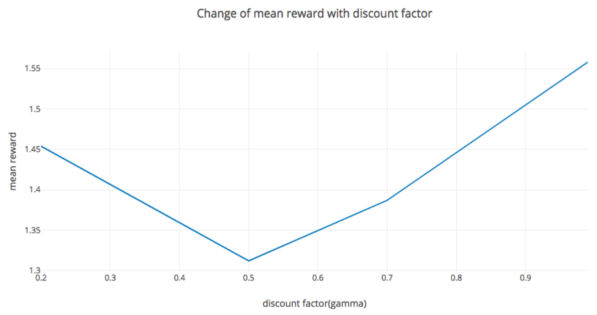
1.1 Questions to be answered:

1. What parts have you implemented and their role in training the agent??
   3 parts have been implemented:
   Keras Sequential Model We have used a deep learning with 2 hidden layers with 128 nodes each and activation from RELU-¿RELU-¿LINEAR.
   Epsilon decay function To balance out exploration and exploitation in reinforcement learning, epsilon is used for the agent to go rogue at first to explore all the possible steps it can take and decay is used to gradually decrease the epsilon value so that it can now learn and take steps on its own decisions.
   Q learning function Q-learning is a value-based reinforcement learning algorithm which is used to find the optimal action selection policy using a Q function. Q table helps us to find the best action for each state. This function can be estimated using Q-learning and after some iterations/episodes, the agent learns and starts to exploit the environment and take better decisions.

2. Can these snippets be improved and how it will influence the training the agent?
   All the snippets take some hyper-parameters like gamma, activation function, learning rate and epsilon which can be changed to improve the performance of the system.
   Change with number of episodes:
   An episode is a complete play from one of the initial state to a final state. We see the more the no. of episodes are played the mean reward increases and the time to learn decreases as the model is more trained though after a certain no of episodes, the change in mean reward goes minimal.
   Change with gamma



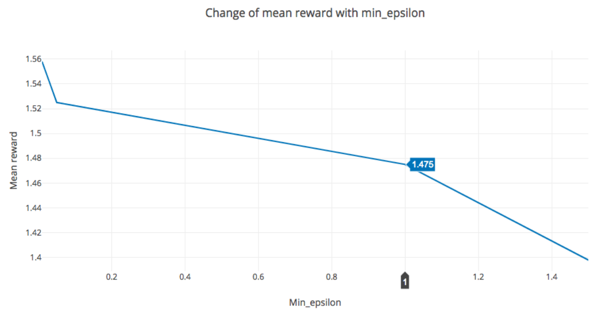
The discount factor is a measure of how far ahead in time the algorithm



looks.A discount factor closer to zero on the other hand indicates that only rewards in the immediate future are being considered, implying a shallow lookahead. We see in the above figure as the discount factor inches towards 1, the mean reward value increases.

Change with min-epsilon

With the decrease in the min-epsilon value, the difference between the



max and min epsilon increases thereby increasing the value of decay and the agent is able to explore for a little while and starts acting greedily shortly after and therefore the mean reward goes down.

3. How quickly your agent were able to learn?
   With the optimizer= RMSprop, LR= 0.00025, GAMMA-0.99 the time elapsed per iteration is 1544052760.57s. However, by changing different hyperparameters we have observed significant change in the time elapsed to train the agent.

2 Writing Part

2.1 Explain what happens in reinforcement learning if the agent always chooses the action that maximizes the Q-value. Suggest two ways to force the agent to explore.

When an agent always chooses the action that maximizes the Q-value, the exploration versus exploitation aspect of the reinforcement learning comes into play, that is the agent only exploits the moves with the maximum reward instead of exploring all the moves that it can take to achieve an even bigger reward covering the shortest distance.The technique in other words is known as the **GREEDY POLICY** where at any given time , the agent chooses the action which it expects to provide the greatest reward. The problem with a greedy approach is that it almost universally arrives at a sub-optimal solution.
Consider a simple example where there are two arms with rewards.If we suppose one arm gives a reward of 1 and the other arm gives a reward of 2, then if the agent's parameters are such that it chooses the former arm first, then regardless of how complex a neural network we utilize, under a greedy approach it will never learn that the latter action is more optimal.
There are some popular approaches to force the agent to explore:

1. -Greedy Approach
   The opposite to greedy approach is random approach where the agent always takes a random approach which has its own shortcomings. Therefore, a combination of greedy and random approach seems to be effective. In this approach the agent chooses what it believes to be the optimal action most of the time, but occasionally acts randomly. The in -greedy is an adjustable parameter which determines the probability of taking a random, rather than principled, action.

2. Bayesian Approaches (w/ Dropout)
   Dropout is a technique in neural networks where the network activations are randomly set to zero to avoid over-fitting.Greedy policy can be seen as a variation of greedy policy and therefore, the same approach can be taken here. More specifically in a bayesian approach, some probabilistic weights are introduced into the system combined with dropout which brings an uncertainty about each action.

There are some other advanced techniques like intrinsic motivation which can be involved to force the agent to explore.

2.2 Q-table for the given steps

$$Q(st, at) = rt + maxaQ(st + 1, a) \quad (1)$$

Actions/States	Up	Down	Left	Right
0	3.90	3.94	3.90	3.94
1	2.94	2.97	2.90	2.97
2	1.94	1.99	1.94	1.99
3	0.97	1	0.97	0.99
4	0	0	0	0

2.2.1 Calculation

States: S0, S1, S2, S3, S4
Actions: up, down, left and right
At state S4,
No more values can be taken as the agent is at terminal step
Therefore,Q(S4,right) = Q(s4,down) = Q(s4,left) = Q(S4,up) = 0
At state S3,
The optimal path is to move down
Q(S3,down) = 1 + 0.99(max(s4))= 1+0.99(0)= 1
Q(S3,right) = 1 + 0.99(max(s3)) = 0+0.99(1) = 0.99
Q(S3,left) = -1 + 0.99(max(s2)) = -1+0.99(1.99) = 0.97
Q(S3,up) = -1 + 0.99(max(S2)) = -1+0.99(1.99) = 0.97
At state S2,
Q(S2,right) = 1+0.99(max(S3)) = 1+0.99(1) = 1.99
Q(S2,down) = 1+0.99(max(S3)) = 1+0.99(1) = 1.99
Q(s2,left) = -1+0.99(ma(s1)) = -1 + 0.99(2.97) = 1.94
Q(S2,up) = -1 + 0.99(max(s1)) = -1 + 0.99(2.97) = 1.94
At state S1,
Q(S1,right) = 1+0.99(max(s2)) = 1+0.99(1.99) = 2.97
Q(S1,down) = 1+0.99(max(s2)) = 1+0.99(1.99) = 2.97
Q(S1,left) = -1 + 0.99(max(s0)) = -1+0.99(3.94) = 2.90
Q(S1,up) = 0 + 0.99(max(S1)) = 0+0.99(2.97) = 2.94
At state S0,
Q(S0,right) = 1+0.99(max(s1)) = 1+0.99(2.97) = 3.94
Q(S0,down) = 1+0.99(max(s1)) = 1+0.99(2.97) = 3.94
Q(S0,left) = 0+0.99(max(s0)) = 1+0.99(3.94) = 3.90
Q(S0,up) = 0+0.99(max(s0)) = 1+0.99(3.94) = 3.90

3 References

https://medium.com/@m.alzantot/deep-reinforcement-learning-demysitifed-episode-2-policy-iteration-value-iteration-and-q-978f9e89ddaa