Exploring Deep Q Network for Atari Game

Sumanyu Garg
Data Science
Indiana University
Bloomington, IN 47408
sumgarg@iu.edu

Abstract

In this project, I experiment with the Deep Q Networks on Atari Environment. These networks are able to learn policies from the input using reinforcement Learning. The network is trained with a variant of Q-learning, with input as raw pixels from the screen and the output is action value function which estimates future rewards for each action.

1 Introduction

Learning control policies directly from images or some other high dimensional sensory input was not possible before the advent of deep learning.

However, even with deep learning at our disposal, there are several challenges in applying to solve reinforcement learning problems. Deep Learning methods require large amounts of training data and how that can be mapped to a reinforcement learning problem is not immediately clear. One might argue to learn a model of the environment by estimating transition probabilities and reward functions but that is almost impossible for most environments due to very large state space and stochasticity in the system. Another issue with deep learning-based methods is that we need our training samples to be independent whereas in reinforcement learning, we get a sequence of states which have high correlation between them. Deep Q Networks [1], along with a variant of Q-learning [2] helps us to tackle these challenges.

2 Background

In this project, I will be experimenting with Atari environment [3], specifically the *SpaceInvaders-v0'* environment.(However, the implementation is well organized to be able to tackle other environments as well with minor modifications that will be needed as per the dynamics of different environments.)
In general, any reinforcement learning problem with single agent consists of an environment, and at each time step, the agent selects an action a_t from the agent's action space. (There are 6 actions in *SpaceInvaders-v0* environment, *{0:no action}, {1:fire}, {2:move_right}, {3:move_left}, {4:move_right_fire}, {5:move_left_fire}).* The agent gets to only observe the images of the current screen x_t.

The goal of the agent is to interact with the environment by selecting actions in a way that maximizes future rewards. I have considered discounted rewards such that the discounted *return* at time *t* can be written as $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ where *T* is the time-step at which the game ends.
We can also define the optimal action value function, which can be defined as the maximum expected reward achievable by following an optimal policy, using *Bellman equation* as follows.

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q^*(s', a') \Big| s, a \right]$$

In this project, this action value function is estimated using a neural network as a function approximator. We can use one neural network for each action or alternatively use one single neural network that will approximate the action value function for each action.

$$Q(s, a; \theta) \approx Q^*(s, a).$$

This network can be trained by minimizing a sequence of loss functions $L_i(\theta_i)$ that changes at each iteration i.

$$L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot)} \left[(y_i - Q(s, a; \theta_i))^2 \right] ,$$

where $y_i = \mathbb{E}_{s' \sim \mathcal{E}} [r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) | s, a]$ is the target for iteration i and $\rho(s, a)$ is a probability distribution over sequences *s* and actions *a* that the authors of the original paper referred to as *behaviour distribution.* One important thing to note is that target depends on the network weights which is in contrast with what the target is in supervised learning, which are fixed for all iterations during training. Now, if we differentiate the loss function with respect to the weights, then the gradient can be written as follows.

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot); s' \sim \mathcal{E}} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i) \right) \nabla_{\theta_i} Q(s, a; \theta_i) \right]$$

However, how can we compute these gradients practically? It is computationally infeasible to calculate the full gradients and hence we use stochastic gradient descent to compute them This algorithm is same as the *Q-Learning* algorithm. Some important things to note are that this algorithm as *model-free*, which means that we don't estimate the dynamics of the environment and instead solve directly from the samples. Another thing to note is that this algorithm is *off-policy*, which means it learns an optimal policy by following a behaviour distribution. Now how to select a behaviour distribution? In practice, behaviour distribution is chosen to be an ϵ-greedy strategy which allows exploration by choosing the optimal action with probability 1- ϵ and random action by probability ϵ.

3 Deep Reinforcement Learning

Deep Reinforcement Learning uses something called as *experience replay* to store the agent's experience at each time step, $e_t = (s_t, a_t, r_t, s_{t+1})$ in a dataset $\mathcal{D} = e_1, ..., e_N$ during each episode. The weights of the network are updated using Q-learning updates using samples of experience, $e \sim \mathcal{D}$, drawn at random from the experience replay dataset. After performing experience replay, the agent selects and executes action according to ϵ-greedy strategy. This allows for the experiences to be used in many weight updates and allows for greater data efficiency. Moreover, the samples are not correlated as the correlation is broken due to randomized sampling which acts as a training dataset for updating the network. This in turn reduces the variances of the updates. The most important thing is that, when learning on-policy, the current parameters determine the next data sample that the parameters are trained on. Due to this, it is possible that the parameters might stuck in some local minima and do not converge to the desired optimal solution. Therefore, we use off-policy learning, which in fact becomes necessary due to the use of experience replay to update the weights and not the correlated samples. The diagram below, taken from the original paper, shows the complete algorithm.

Algorithm 1 Deep Q-learning with Experience Replay

```
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights
for episode = 1, M do
    Initialise sequence s_1 = {x_1} and preprocessed sequenced φ_1 = φ(s_1)
    for t = 1, T do
        With probability ε select a random action a_t
        otherwise select a_t = max_a Q*(φ(s_t), a; θ)
        Execute action a_t in emulator and observe reward r_t and image x_{t+1}
        Set s_{t+1} = s_t, a_t, x_{t+1} and preprocess φ_{t+1} = φ(s_{t+1})
        Store transition (φ_t, a_t, r_t, φ_{t+1}) in D
        Sample random minibatch of transitions (φ_j, a_j, r_j, φ_{j+1}) from D
        Set y_j = { r_j                                  for terminal φ_{j+1}
                  { r_j + γ max_{a'} Q(φ_{j+1}, a'; θ)   for non-terminal φ_{j+1}
        Perform a gradient descent step on (y_j − Q(φ_j, a_j; θ))^2 according to equation 3
    end for
end for
```

4 Pre-Processing and Model Architecture

OpenAI gym environment for Atari games returns a RGB image as the state of the environment. However, we don't really need colored frames to capture the information in those frames. Therefore, each frame is converted into grayscale as the first preprocessing step. Cropping of the frame, to get a 185x95 size frame, is also done in order to capture only the relevant information.

The network takes input as the processed image and outputs the action value function for each action. The network architecture consists of one convolution layer and 2 linear layers due to limited computational resources (however, one can experiment with a much more complex network in order to get much better results). The first layer is a *convolutional layer* with 64 output channels, each kernel of size 3x3 and stride as 1. The second layer is a *linear layer* with 64 neurons. The final layer is the output layer with number of outputs as the number of actions, the agent can perform (6 in the case of '*SpaceInvaders*').

5 Experiments

I have conducted experiments only on the 'SpaceInvaders' environment due to limited computational resources and strict timeline. However, the network architecture is robust enough to be used for other games as well. The original paper scales all the positive rewards to be 1 and all negative rewards to be -1. However, in my implementation, I haven't done so. I wanted to experiment with the original reward setting instead of modifying it. However, one important change that I have done, is to give a large negative reward when the episode ends. This will avoid the agent to actions which lead to end of the games or episodes. This has been done only during the learning part of the algorithm. While storing the experience into replay buffer, I have stored the original rewards without modifying them.

During the training, I have used the RMSProp algorithm with minibatches of size 32. The behaviour policy during training was ϵ-greedy with ϵ varied exponentially from 0.95 to 0.05 over a course of 500 episodes. Since the number of episodes is very less (restricted to this number only due to limited computational resources), the epsilon decreased only till 0.1. Also, the original paper decreased epsilon across each frame, whereas I have experimented by decreasing epsilon for each episode.

I have also used a frame-skipping technique which is generally used for Atari games. More precisely, the agent selects actions on every k^{th} frame instead of every frame, and the last action is repeated for k skipped frames. I have used $k=3$ which is same as the one used by the original paper.

6 Results

Figure 1 shows the results during training of the network. We can see from the plots that the agent is learning to behave in the environment in order to maximize its total expected reward.

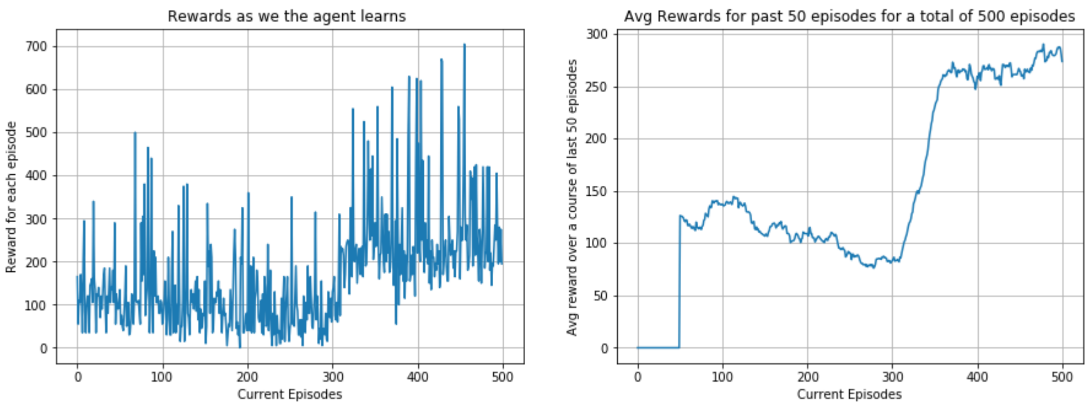


Figure 1: The plot on the left shows total reward per episode on *Space Invaders* environment during training. The plot on the right shows average reward for the past 50 episodes.

The plot on the left in figure 1 shows the total reward that agent is able to get during each episode. The agent manages to achieve a maximum reward of 705 at 456th episode. The agent starts with taking random actions and gradually it starts choosing actions which lead to better states and hence better states. However, we can see lot of randomness in the total reward for an episode. This is due to the extreme stochastic nature of the *SpaceInvaders* environment. In plot on the right-hand side, average rewards for past 50 episodes has been plotted. Initially, I tried to evaluate policy after 10 episodes. But this was computationally expensive. Therefore, I plotted the average reward for the last 50 episodes. Important thing to note here is that the policy is changing with each episode and therefore it is not same as evaluating policy after every few episode. Nonetheless, we can see that the as the number of episodes is increasing, the average reward is also increasing which is indicating that the agent is trying to learn the optimal policy.

7 Conclusion

In this project, I explored Deep Q Networks on Atari environment. The agent was able to learn a good enough policy and was taking actions which will lead to better rewards instead of just taking actions randomly. There were many difficulties that I had to face during this project. I read the paper on Deep Q Networks to gain insights of how these networks work. In the process, I also learned about the importance of experience replay and off-policy learning in the case of deep q networks. Theoretical understanding is one thing and actually implementing is another. I faced many problems during implementation, mainly on how to divide the code into different sections to make it self-explanatory and extendable to other environments as well. If I were to start this project from scratch, I would try to extend my work to use deep double Q learning [3] which is an extension of double Q Learning using neural networks. I will also try to compare the results of different algorithms across different environments. I was unable to do this in this project due to limited resources and a time constraint. But I would surely like to try out these in future.

8 References

[1] *Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller*. Playing Atari with Deep Reinforcement Learning.

[2] *Christopher JCH Watkins and Peter Dayan*. Q-learning. Machine learning, 8(3-4):279–292, 1992

[3] *Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, Wojciech Zaremba.* OpenAI Gym

[4] *Hado van Hasselt, Arthur Guez, David Silver*. Deep Reinforcement Learning with Double Q-Learning.

DeepQNetwork

April 30, 2020

[1]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

import numpy as np
import math
import random

import gym
```

[2]:
```
class DeepQNetwork(nn.Module):
    def __init__(self, lr, img_size, num_actions):
        super(DeepQNetwork, self).__init__()
        self.img_size = img_size
        self.num_actions = num_actions
        self.lr = lr
        self.momentum = 0.9

        self.conv1 = nn.Conv2d(in_channels = 1, out_channels = 64, kernel_size␣
↪= 3, stride = 1)
        in_features = self.calculate_dim_fc_layer()
        self.fc1 = nn.Linear(in_features = in_features, out_features = 64)
        self.fc2 = nn.Linear(in_features = 64, out_features = self.num_actions)␣
↪# There are 6 actions.

        self.optimizer = optim.RMSprop(self.parameters(), lr = self.lr,␣
↪momentum = self.momentum)
        self.loss = nn.MSELoss()
        self.device = torch.device('cuda:0' if torch.cuda.is_available() else␣
↪'cpu')
        self.to(self.device)
```

```python
    def calculate_dim_fc_layer(self):
        state = torch.zeros(1, *self.img_size)
        dims = self.conv1(state)
        return int(np.prod(dims.size()))

    def forward(self, observation):
        observation = torch.Tensor(observation).to(self.device)   # Observation is Heigh, Width , Channels
        observation =  observation.view(-1, 1, self.img_size[1], self.img_size[2]) # However we want channels to come first as we can see in convolution layer.
                                                                  # -1 will take care of the number of frames we are passing.
        observation = F.relu(self.conv1(observation))
        observation = observation.flatten(start_dim = 1)
        observation = F.relu(self.fc1(observation))
        actions = self.fc2(observation)
        return actions
```

[ ]:

```python

```

[ ]:

```python

```

DeepQNetwork

April 30, 2020

[1]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

import numpy as np
import math
import random

import gym
```

[2]:
```
class DeepQNetwork(nn.Module):
    def __init__(self, lr, img_size, num_actions):
        super(DeepQNetwork, self).__init__()
        self.img_size = img_size
        self.num_actions = num_actions
        self.lr = lr
        self.momentum = 0.9

        self.conv1 = nn.Conv2d(in_channels = 1, out_channels = 64, kernel_size␣
↪= 3, stride = 1)
        in_features = self.calculate_dim_fc_layer()
        self.fc1 = nn.Linear(in_features = in_features, out_features = 64)
        self.fc2 = nn.Linear(in_features = 64, out_features = self.num_actions)␣
↪# There are 6 actions.

        self.optimizer = optim.RMSprop(self.parameters(), lr = self.lr,␣
↪momentum = self.momentum)
        self.loss = nn.MSELoss()
        self.device = torch.device('cuda:0' if torch.cuda.is_available() else␣
↪'cpu')
        self.to(self.device)
```

```
    def calculate_dim_fc_layer(self):
        state = torch.zeros(1, *self.img_size)
        dims = self.conv1(state)
        return int(np.prod(dims.size()))

    def forward(self, observation):
        observation = torch.Tensor(observation).to(self.device)   # Observation is Heigh, Width , Channels
        observation =  observation.view(-1, 1, self.img_size[1], self.img_size[2]) # However we want channels to come first as we can see in convolution layer.
                                                                                   # -1 will take care of the number of frames we are passing.
        observation = F.relu(self.conv1(observation))
        observation = observation.flatten(start_dim = 1)
        observation = F.relu(self.fc1(observation))
        actions = self.fc2(observation)
        return actions
```

[ ]:

[ ]:

Main

April 30, 2020

0.1 Importing Libraries

[44]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

import numpy as np
import math
import random

import gym

from DeepQNetwork import DeepQNetwork
from Agent import Agent

from tqdm.notebook import tqdm
import matplotlib.pyplot as plt
```

0.2 Utility Functions for calculating average reward and plotting graphs

[72]:
```
def calculate_avg_reward(scores, k):
    avg_scores = []
    running_sum = 0
    for i in range(k):
        running_sum+=scores[i]
        avg_scores.append(0)

    avg_scores.append(running_sum)

    for i in range(k,episodes,1):
        new_avg_score = avg_scores[-1] - scores[i-k] + scores[i]
        avg_scores.append(new_avg_score)

    avg_scores = np.array(avg_scores)/k
    return avg_scores
```

```
def plot_graphs(scores, avg_scores):
    plt.figure(1, figsize = (15,5))

    plt.subplot(121)
    plt.plot(scores)
    plt.xlabel('Current Episodes')
    plt.ylabel('Reward for each episode')
    plt.title('Rewards as we the agent learns ')
    plt.grid()

    plt.subplot(122)
    plt.plot(avg_scores)
    plt.xlabel('Current Episodes')
    plt.ylabel('Avg reward over a course of last 50 episodes')
    plt.title('Avg Rewards for past 50 episodes for a total of 500 episodes')
    plt.grid()
```

0.3 Main Program

(i) Defining some variables

[45]:
```
env = gym.make('SpaceInvaders-v0')
num_actions = 6 # 0 no action, 1 fire, 2 move right, 3 move left, 4 move right␣
↪fire, 5 move left fire
scores = []
episodes = 500
batch_size = 32
```

(ii) Making an object of agent class and initialising Experience Replay memory with random transitions

[ ]:
```
agent = Agent(num_actions)
```

[46]:
```
while agent.memCntr < agent.memSize:
    state = env.reset()
    done = False
    while not done:
        action = env.action_space.sample()
        next_state, reward, done, info = env.step(action)
        if done and info['ale.lives'] == 0: # TO avoid agent to loose, we are␣
↪giving high penalty
            reward = -50
        agent.storeTransition(agent.process_state(state), action, reward, agent.
↪process_state(next_state))
        state = next_state
print('done initializing memory')
```

```
(1, 185, 95)
(1, 185, 95)
done initializing memory
```

(iii) Main loop

[10]:
```python
for i in tqdm(range(episodes)):
    print('starting episode ', i+1, 'epsilon: %.4f' % agent.epsilon)
    done = False
    state = env.reset()
    frames = [agent.process_state(state)]
    score = 0
    lastAction = 0
    agent.updateEpsilon(i)
    while not done: # Action is repeated for 3 frames.
        if len(frames) == 3:
            action = agent.chooseAction(frames)
            frames = []
        else:
            action = lastAction
        next_state, reward, done, info = env.step(action)
        score += reward
        frames.append(agent.process_state(next_state))
        if done and info['ale.lives'] == 0:
            reward = -50
        agent.storeTransition(agent.process_state(state), action, reward, agent.
 ↪process_state(next_state))
        state = next_state
        agent.learn(batch_size)
        lastAction = action
    scores.append(score)
    print('score:',score)
```

```
HBox(children=(FloatProgress(value=0.0, max=500.0), HTML(value='')))


starting episode  1 epsilon: 0.9500
score: 165.0
starting episode  2 epsilon: 0.9500
score: 55.0
starting episode  3 epsilon: 0.9455
score: 110.0
starting episode  4 epsilon: 0.9410
score: 105.0
starting episode  5 epsilon: 0.9366
score: 170.0
starting episode  6 epsilon: 0.9322
score: 125.0
starting episode  7 epsilon: 0.9278
```

```
score: 35.0
starting episode  8 epsilon: 0.9234
score: 205.0
starting episode  9 epsilon: 0.9190
score: 295.0
starting episode  10 epsilon: 0.9147
score: 85.0
starting episode  11 epsilon: 0.9104
score: 35.0
starting episode  12 epsilon: 0.9061
score: 95.0
starting episode  13 epsilon: 0.9018
score: 120.0
starting episode  14 epsilon: 0.8976
score: 120.0
starting episode  15 epsilon: 0.8934
score: 35.0
starting episode  16 epsilon: 0.8892
score: 145.0
starting episode  17 epsilon: 0.8850
score: 150.0
starting episode  18 epsilon: 0.8808
score: 160.0
starting episode  19 epsilon: 0.8767
score: 105.0
starting episode  20 epsilon: 0.8725
score: 340.0
starting episode  21 epsilon: 0.8684
score: 160.0
starting episode  22 epsilon: 0.8644
score: 135.0
starting episode  23 epsilon: 0.8603
score: 35.0
starting episode  24 epsilon: 0.8563
score: 125.0
starting episode  25 epsilon: 0.8522
score: 120.0
starting episode  26 epsilon: 0.8482
score: 140.0
starting episode  27 epsilon: 0.8442
score: 110.0
starting episode  28 epsilon: 0.8403
score: 70.0
starting episode  29 epsilon: 0.8363
score: 120.0
starting episode  30 epsilon: 0.8324
score: 90.0
starting episode  31 epsilon: 0.8285
```

```
score: 130.0
starting episode  32 epsilon: 0.8246
score: 170.0
starting episode  33 epsilon: 0.8208
score: 185.0
starting episode  34 epsilon: 0.8169
score: 90.0
starting episode  35 epsilon: 0.8131
score: 35.0
starting episode  36 epsilon: 0.8093
score: 120.0
starting episode  37 epsilon: 0.8055
score: 80.0
starting episode  38 epsilon: 0.8017
score: 135.0
starting episode  39 epsilon: 0.7980
score: 185.0
starting episode  40 epsilon: 0.7943
score: 105.0
starting episode  41 epsilon: 0.7906
score: 140.0
starting episode  42 epsilon: 0.7869
score: 135.0
starting episode  43 epsilon: 0.7832
score: 180.0
starting episode  44 epsilon: 0.7795
score: 105.0
starting episode  45 epsilon: 0.7759
score: 290.0
starting episode  46 epsilon: 0.7723
score: 55.0
starting episode  47 epsilon: 0.7687
score: 125.0
starting episode  48 epsilon: 0.7651
score: 105.0
starting episode  49 epsilon: 0.7615
score: 90.0
starting episode  50 epsilon: 0.7580
score: 95.0
starting episode  51 epsilon: 0.7544
score: 135.0
starting episode  52 epsilon: 0.7509
score: 55.0
starting episode  53 epsilon: 0.7474
score: 75.0
starting episode  54 epsilon: 0.7439
score: 50.0
starting episode  55 epsilon: 0.7405
```

```
score: 40.0
starting episode  56 epsilon: 0.7370
score: 80.0
starting episode  57 epsilon: 0.7336
score: 120.0
starting episode  58 epsilon: 0.7302
score: 190.0
starting episode  59 epsilon: 0.7268
score: 50.0
starting episode  60 epsilon: 0.7234
score: 75.0
starting episode  61 epsilon: 0.7201
score: 105.0
starting episode  62 epsilon: 0.7167
score: 30.0
starting episode  63 epsilon: 0.7134
score: 35.0
starting episode  64 epsilon: 0.7101
score: 65.0
starting episode  65 epsilon: 0.7068
score: 125.0
starting episode  66 epsilon: 0.7035
score: 105.0
starting episode  67 epsilon: 0.7003
score: 55.0
starting episode  68 epsilon: 0.6970
score: 110.0
starting episode  69 epsilon: 0.6938
score: 500.0
starting episode  70 epsilon: 0.6906
score: 110.0
starting episode  71 epsilon: 0.6874
score: 105.0
starting episode  72 epsilon: 0.6842
score: 105.0
starting episode  73 epsilon: 0.6811
score: 110.0
starting episode  74 epsilon: 0.6779
score: 80.0
starting episode  75 epsilon: 0.6748
score: 55.0
starting episode  76 epsilon: 0.6717
score: 290.0
starting episode  77 epsilon: 0.6686
score: 155.0
starting episode  78 epsilon: 0.6655
score: 305.0
starting episode  79 epsilon: 0.6624
```

```
score: 170.0
starting episode  80 epsilon: 0.6594
score: 380.0
starting episode  81 epsilon: 0.6563
score: 75.0
starting episode  82 epsilon: 0.6533
score: 110.0
starting episode  83 epsilon: 0.6503
score: 135.0
starting episode  84 epsilon: 0.6473
score: 465.0
starting episode  85 epsilon: 0.6443
score: 180.0
starting episode  86 epsilon: 0.6413
score: 35.0
starting episode  87 epsilon: 0.6384
score: 105.0
starting episode  88 epsilon: 0.6355
score: 440.0
starting episode  89 epsilon: 0.6325
score: 155.0
starting episode  90 epsilon: 0.6296
score: 35.0
starting episode  91 epsilon: 0.6267
score: 225.0
starting episode  92 epsilon: 0.6239
score: 120.0
starting episode  93 epsilon: 0.6210
score: 210.0
starting episode  94 epsilon: 0.6182
score: 105.0
starting episode  95 epsilon: 0.6153
score: 105.0
starting episode  96 epsilon: 0.6125
score: 120.0
starting episode  97 epsilon: 0.6097
score: 80.0
starting episode  98 epsilon: 0.6069
score: 80.0
starting episode  99 epsilon: 0.6041
score: 110.0
starting episode  100 epsilon: 0.6014
score: 105.0
starting episode  101 epsilon: 0.5986
score: 55.0
starting episode  102 epsilon: 0.5959
score: 65.0
starting episode  103 epsilon: 0.5932
```

```
score: 105.0
starting episode  104 epsilon: 0.5904
score: 155.0
starting episode  105 epsilon: 0.5878
score: 110.0
starting episode  106 epsilon: 0.5851
score: 30.0
starting episode  107 epsilon: 0.5824
score: 65.0
starting episode  108 epsilon: 0.5797
score: 210.0
starting episode  109 epsilon: 0.5771
score: 105.0
starting episode  110 epsilon: 0.5745
score: 30.0
starting episode  111 epsilon: 0.5719
score: 30.0
starting episode  112 epsilon: 0.5693
score: 180.0
starting episode  113 epsilon: 0.5667
score: 270.0
starting episode  114 epsilon: 0.5641
score: 35.0
starting episode  115 epsilon: 0.5615
score: 50.0
starting episode  116 epsilon: 0.5590
score: 145.0
starting episode  117 epsilon: 0.5564
score: 35.0
starting episode  118 epsilon: 0.5539
score: 100.0
starting episode  119 epsilon: 0.5514
score: 55.0
starting episode  120 epsilon: 0.5489
score: 330.0
starting episode  121 epsilon: 0.5464
score: 145.0
starting episode  122 epsilon: 0.5439
score: 15.0
starting episode  123 epsilon: 0.5415
score: 50.0
starting episode  124 epsilon: 0.5390
score: 50.0
starting episode  125 epsilon: 0.5366
score: 110.0
starting episode  126 epsilon: 0.5341
score: 375.0
starting episode  127 epsilon: 0.5317
```

```
score: 15.0
starting episode  128 epsilon: 0.5293
score: 50.0
starting episode  129 epsilon: 0.5269
score: 140.0
starting episode  130 epsilon: 0.5246
score: 380.0
starting episode  131 epsilon: 0.5222
score: 80.0
starting episode  132 epsilon: 0.5198
score: 70.0
starting episode  133 epsilon: 0.5175
score: 30.0
starting episode  134 epsilon: 0.5152
score: 105.0
starting episode  135 epsilon: 0.5128
score: 135.0
starting episode  136 epsilon: 0.5105
score: 105.0
starting episode  137 epsilon: 0.5082
score: 125.0
starting episode  138 epsilon: 0.5060
score: 90.0
starting episode  139 epsilon: 0.5037
score: 125.0
starting episode  140 epsilon: 0.5014
score: 155.0
starting episode  141 epsilon: 0.4992
score: 85.0
starting episode  142 epsilon: 0.4969
score: 165.0
starting episode  143 epsilon: 0.4947
score: 65.0
starting episode  144 epsilon: 0.4925
score: 135.0
starting episode  145 epsilon: 0.4903
score: 35.0
starting episode  146 epsilon: 0.4881
score: 90.0
starting episode  147 epsilon: 0.4859
score: 45.0
starting episode  148 epsilon: 0.4837
score: 50.0
starting episode  149 epsilon: 0.4816
score: 80.0
starting episode  150 epsilon: 0.4794
score: 75.0
starting episode  151 epsilon: 0.4773
```

```
score: 155.0
starting episode  152 epsilon: 0.4751
score: 10.0
starting episode  153 epsilon: 0.4730
score: 50.0
starting episode  154 epsilon: 0.4709
score: 335.0
starting episode  155 epsilon: 0.4688
score: 190.0
starting episode  156 epsilon: 0.4667
score: 225.0
starting episode  157 epsilon: 0.4646
score: 80.0
starting episode  158 epsilon: 0.4626
score: 240.0
starting episode  159 epsilon: 0.4605
score: 210.0
starting episode  160 epsilon: 0.4585
score: 35.0
starting episode  161 epsilon: 0.4564
score: 50.0
starting episode  162 epsilon: 0.4544
score: 115.0
starting episode  163 epsilon: 0.4524
score: 105.0
starting episode  164 epsilon: 0.4504
score: 105.0
starting episode  165 epsilon: 0.4484
score: 130.0
starting episode  166 epsilon: 0.4464
score: 120.0
starting episode  167 epsilon: 0.4444
score: 155.0
starting episode  168 epsilon: 0.4424
score: 80.0
starting episode  169 epsilon: 0.4405
score: 135.0
starting episode  170 epsilon: 0.4385
score: 80.0
starting episode  171 epsilon: 0.4366
score: 35.0
starting episode  172 epsilon: 0.4347
score: 80.0
starting episode  173 epsilon: 0.4328
score: 115.0
starting episode  174 epsilon: 0.4308
score: 75.0
starting episode  175 epsilon: 0.4289
```

```
score: 80.0
starting episode  176 epsilon: 0.4271
score: 50.0
starting episode  177 epsilon: 0.4252
score: 5.0
starting episode  178 epsilon: 0.4233
score: 30.0
starting episode  179 epsilon: 0.4214
score: 55.0
starting episode  180 epsilon: 0.4196
score: 50.0
starting episode  181 epsilon: 0.4177
score: 135.0
starting episode  182 epsilon: 0.4159
score: 90.0
starting episode  183 epsilon: 0.4141
score: 40.0
starting episode  184 epsilon: 0.4123
score: 120.0
starting episode  185 epsilon: 0.4105
score: 220.0
starting episode  186 epsilon: 0.4087
score: 275.0
starting episode  187 epsilon: 0.4069
score: 210.0
starting episode  188 epsilon: 0.4051
score: 45.0
starting episode  189 epsilon: 0.4033
score: 60.0
starting episode  190 epsilon: 0.4016
score: 30.0
starting episode  191 epsilon: 0.3998
score: 50.0
starting episode  192 epsilon: 0.3981
score: 0.0
starting episode  193 epsilon: 0.3963
score: 210.0
starting episode  194 epsilon: 0.3946
score: 195.0
starting episode  195 epsilon: 0.3929
score: 325.0
starting episode  196 epsilon: 0.3912
score: 35.0
starting episode  197 epsilon: 0.3895
score: 35.0
starting episode  198 epsilon: 0.3878
score: 45.0
starting episode  199 epsilon: 0.3861
```

```
score: 75.0
starting episode  200 epsilon: 0.3844
score: 80.0
starting episode  201 epsilon: 0.3828
score: 40.0
starting episode  202 epsilon: 0.3811
score: 360.0
starting episode  203 epsilon: 0.3794
score: 105.0
starting episode  204 epsilon: 0.3778
score: 40.0
starting episode  205 epsilon: 0.3762
score: 190.0
starting episode  206 epsilon: 0.3745
score: 35.0
starting episode  207 epsilon: 0.3729
score: 80.0
starting episode  208 epsilon: 0.3713
score: 210.0
starting episode  209 epsilon: 0.3697
score: 35.0
starting episode  210 epsilon: 0.3681
score: 115.0
starting episode  211 epsilon: 0.3665
score: 130.0
starting episode  212 epsilon: 0.3649
score: 180.0
starting episode  213 epsilon: 0.3634
score: 155.0
starting episode  214 epsilon: 0.3618
score: 70.0
starting episode  215 epsilon: 0.3603
score: 60.0
starting episode  216 epsilon: 0.3587
score: 70.0
starting episode  217 epsilon: 0.3572
score: 110.0
starting episode  218 epsilon: 0.3556
score: 35.0
starting episode  219 epsilon: 0.3541
score: 125.0
starting episode  220 epsilon: 0.3526
score: 30.0
starting episode  221 epsilon: 0.3511
score: 165.0
starting episode  222 epsilon: 0.3496
score: 140.0
starting episode  223 epsilon: 0.3481
```

```
score: 50.0
starting episode  224 epsilon: 0.3466
score: 55.0
starting episode  225 epsilon: 0.3451
score: 240.0
starting episode  226 epsilon: 0.3437
score: 40.0
starting episode  227 epsilon: 0.3422
score: 85.0
starting episode  228 epsilon: 0.3407
score: 180.0
starting episode  229 epsilon: 0.3393
score: 35.0
starting episode  230 epsilon: 0.3378
score: 5.0
starting episode  231 epsilon: 0.3364
score: 80.0
starting episode  232 epsilon: 0.3350
score: 30.0
starting episode  233 epsilon: 0.3336
score: 130.0
starting episode  234 epsilon: 0.3321
score: 55.0
starting episode  235 epsilon: 0.3307
score: 5.0
starting episode  236 epsilon: 0.3293
score: 75.0
starting episode  237 epsilon: 0.3279
score: 35.0
starting episode  238 epsilon: 0.3266
score: 40.0
starting episode  239 epsilon: 0.3252
score: 10.0
starting episode  240 epsilon: 0.3238
score: 10.0
starting episode  241 epsilon: 0.3224
score: 115.0
starting episode  242 epsilon: 0.3211
score: 20.0
starting episode  243 epsilon: 0.3197
score: 115.0
starting episode  244 epsilon: 0.3184
score: 165.0
starting episode  245 epsilon: 0.3170
score: 15.0
starting episode  246 epsilon: 0.3157
score: 85.0
starting episode  247 epsilon: 0.3144
```

```
score: 120.0
starting episode  248 epsilon: 0.3131
score: 165.0
starting episode  249 epsilon: 0.3118
score: 50.0
starting episode  250 epsilon: 0.3104
score: 15.0
starting episode  251 epsilon: 0.3091
score: 45.0
starting episode  252 epsilon: 0.3079
score: 60.0
starting episode  253 epsilon: 0.3066
score: 350.0
starting episode  254 epsilon: 0.3053
score: 55.0
starting episode  255 epsilon: 0.3040
score: 60.0
starting episode  256 epsilon: 0.3027
score: 50.0
starting episode  257 epsilon: 0.3015
score: 115.0
starting episode  258 epsilon: 0.3002
score: 190.0
starting episode  259 epsilon: 0.2990
score: 100.0
starting episode  260 epsilon: 0.2977
score: 85.0
starting episode  261 epsilon: 0.2965
score: 30.0
starting episode  262 epsilon: 0.2953
score: 30.0
starting episode  263 epsilon: 0.2941
score: 65.0
starting episode  264 epsilon: 0.2928
score: 25.0
starting episode  265 epsilon: 0.2916
score: 65.0
starting episode  266 epsilon: 0.2904
score: 5.0
starting episode  267 epsilon: 0.2892
score: 120.0
starting episode  268 epsilon: 0.2880
score: 50.0
starting episode  269 epsilon: 0.2868
score: 35.0
starting episode  270 epsilon: 0.2857
score: 105.0
starting episode  271 epsilon: 0.2845
```

```
score: 65.0
starting episode  272 epsilon: 0.2833
score: 190.0
starting episode  273 epsilon: 0.2822
score: 80.0
starting episode  274 epsilon: 0.2810
score: 80.0
starting episode  275 epsilon: 0.2798
score: 165.0
starting episode  276 epsilon: 0.2787
score: 115.0
starting episode  277 epsilon: 0.2776
score: 50.0
starting episode  278 epsilon: 0.2764
score: 45.0
starting episode  279 epsilon: 0.2753
score: 90.0
starting episode  280 epsilon: 0.2742
score: 125.0
starting episode  281 epsilon: 0.2730
score: 315.0
starting episode  282 epsilon: 0.2719
score: 65.0
starting episode  283 epsilon: 0.2708
score: 120.0
starting episode  284 epsilon: 0.2697
score: 80.0
starting episode  285 epsilon: 0.2686
score: 10.0
starting episode  286 epsilon: 0.2675
score: 45.0
starting episode  287 epsilon: 0.2665
score: 20.0
starting episode  288 epsilon: 0.2654
score: 155.0
starting episode  289 epsilon: 0.2643
score: 5.0
starting episode  290 epsilon: 0.2632
score: 45.0
starting episode  291 epsilon: 0.2622
score: 40.0
starting episode  292 epsilon: 0.2611
score: 30.0
starting episode  293 epsilon: 0.2601
score: 80.0
starting episode  294 epsilon: 0.2590
score: 20.0
starting episode  295 epsilon: 0.2580
```

```
score: 15.0
starting episode  296 epsilon: 0.2569
score: 130.0
starting episode  297 epsilon: 0.2559
score: 95.0
starting episode  298 epsilon: 0.2549
score: 70.0
starting episode  299 epsilon: 0.2539
score: 115.0
starting episode  300 epsilon: 0.2528
score: 60.0
starting episode  301 epsilon: 0.2518
score: 130.0
starting episode  302 epsilon: 0.2508
score: 140.0
starting episode  303 epsilon: 0.2498
score: 165.0
starting episode  304 epsilon: 0.2488
score: 75.0
starting episode  305 epsilon: 0.2478
score: 65.0
starting episode  306 epsilon: 0.2468
score: 105.0
starting episode  307 epsilon: 0.2459
score: 60.0
starting episode  308 epsilon: 0.2449
score: 130.0
starting episode  309 epsilon: 0.2439
score: 310.0
starting episode  310 epsilon: 0.2429
score: 75.0
starting episode  311 epsilon: 0.2420
score: 235.0
starting episode  312 epsilon: 0.2410
score: 225.0
starting episode  313 epsilon: 0.2401
score: 230.0
starting episode  314 epsilon: 0.2391
score: 215.0
starting episode  315 epsilon: 0.2382
score: 140.0
starting episode  316 epsilon: 0.2372
score: 170.0
starting episode  317 epsilon: 0.2363
score: 235.0
starting episode  318 epsilon: 0.2354
score: 245.0
starting episode  319 epsilon: 0.2345
```

```
score: 250.0
starting episode  320 epsilon: 0.2335
score: 245.0
starting episode  321 epsilon: 0.2326
score: 125.0
starting episode  322 epsilon: 0.2317
score: 325.0
starting episode  323 epsilon: 0.2308
score: 325.0
starting episode  324 epsilon: 0.2299
score: 165.0
starting episode  325 epsilon: 0.2290
score: 555.0
starting episode  326 epsilon: 0.2281
score: 215.0
starting episode  327 epsilon: 0.2272
score: 200.0
starting episode  328 epsilon: 0.2263
score: 230.0
starting episode  329 epsilon: 0.2255
score: 240.0
starting episode  330 epsilon: 0.2246
score: 150.0
starting episode  331 epsilon: 0.2237
score: 190.0
starting episode  332 epsilon: 0.2228
score: 230.0
starting episode  333 epsilon: 0.2220
score: 230.0
starting episode  334 epsilon: 0.2211
score: 170.0
starting episode  335 epsilon: 0.2203
score: 230.0
starting episode  336 epsilon: 0.2194
score: 235.0
starting episode  337 epsilon: 0.2186
score: 165.0
starting episode  338 epsilon: 0.2177
score: 525.0
starting episode  339 epsilon: 0.2169
score: 265.0
starting episode  340 epsilon: 0.2161
score: 190.0
starting episode  341 epsilon: 0.2152
score: 195.0
starting episode  342 epsilon: 0.2144
score: 285.0
starting episode  343 epsilon: 0.2136
```

```
score: 480.0
starting episode  344 epsilon: 0.2128
score: 335.0
starting episode  345 epsilon: 0.2120
score: 250.0
starting episode  346 epsilon: 0.2112
score: 415.0
starting episode  347 epsilon: 0.2104
score: 260.0
starting episode  348 epsilon: 0.2096
score: 445.0
starting episode  349 epsilon: 0.2088
score: 205.0
starting episode  350 epsilon: 0.2080
score: 240.0
starting episode  351 epsilon: 0.2072
score: 290.0
starting episode  352 epsilon: 0.2064
score: 235.0
starting episode  353 epsilon: 0.2056
score: 245.0
starting episode  354 epsilon: 0.2048
score: 560.0
starting episode  355 epsilon: 0.2041
score: 230.0
starting episode  356 epsilon: 0.2033
score: 160.0
starting episode  357 epsilon: 0.2025
score: 215.0
starting episode  358 epsilon: 0.2018
score: 220.0
starting episode  359 epsilon: 0.2010
score: 350.0
starting episode  360 epsilon: 0.2003
score: 275.0
starting episode  361 epsilon: 0.1995
score: 200.0
starting episode  362 epsilon: 0.1988
score: 175.0
starting episode  363 epsilon: 0.1980
score: 310.0
starting episode  364 epsilon: 0.1973
score: 200.0
starting episode  365 epsilon: 0.1966
score: 310.0
starting episode  366 epsilon: 0.1958
score: 220.0
starting episode  367 epsilon: 0.1951
```

```
score: 270.0
starting episode  368 epsilon: 0.1944
score: 175.0
starting episode  369 epsilon: 0.1937
score: 185.0
starting episode  370 epsilon: 0.1929
score: 265.0
starting episode  371 epsilon: 0.1922
score: 605.0
starting episode  372 epsilon: 0.1915
score: 205.0
starting episode  373 epsilon: 0.1908
score: 145.0
starting episode  374 epsilon: 0.1901
score: 185.0
starting episode  375 epsilon: 0.1894
score: 295.0
starting episode  376 epsilon: 0.1887
score: 55.0
starting episode  377 epsilon: 0.1880
score: 485.0
starting episode  378 epsilon: 0.1873
score: 275.0
starting episode  379 epsilon: 0.1866
score: 100.0
starting episode  380 epsilon: 0.1860
score: 240.0
starting episode  381 epsilon: 0.1853
score: 170.0
starting episode  382 epsilon: 0.1846
score: 250.0
starting episode  383 epsilon: 0.1839
score: 325.0
starting episode  384 epsilon: 0.1833
score: 155.0
starting episode  385 epsilon: 0.1826
score: 225.0
starting episode  386 epsilon: 0.1819
score: 115.0
starting episode  387 epsilon: 0.1813
score: 215.0
starting episode  388 epsilon: 0.1806
score: 155.0
starting episode  389 epsilon: 0.1800
score: 210.0
starting episode  390 epsilon: 0.1793
score: 490.0
starting episode  391 epsilon: 0.1787
```

```
score: 630.0
starting episode  392 epsilon: 0.1780
score: 155.0
starting episode  393 epsilon: 0.1774
score: 170.0
starting episode  394 epsilon: 0.1768
score: 215.0
starting episode  395 epsilon: 0.1761
score: 170.0
starting episode  396 epsilon: 0.1755
score: 235.0
starting episode  397 epsilon: 0.1749
score: 215.0
starting episode  398 epsilon: 0.1743
score: 120.0
starting episode  399 epsilon: 0.1736
score: 425.0
starting episode  400 epsilon: 0.1730
score: 625.0
starting episode  401 epsilon: 0.1724
score: 220.0
starting episode  402 epsilon: 0.1718
score: 475.0
starting episode  403 epsilon: 0.1712
score: 245.0
starting episode  404 epsilon: 0.1706
score: 200.0
starting episode  405 epsilon: 0.1700
score: 620.0
starting episode  406 epsilon: 0.1694
score: 250.0
starting episode  407 epsilon: 0.1688
score: 435.0
starting episode  408 epsilon: 0.1682
score: 230.0
starting episode  409 epsilon: 0.1676
score: 175.0
starting episode  410 epsilon: 0.1670
score: 290.0
starting episode  411 epsilon: 0.1664
score: 210.0
starting episode  412 epsilon: 0.1659
score: 250.0
starting episode  413 epsilon: 0.1653
score: 195.0
starting episode  414 epsilon: 0.1647
score: 445.0
starting episode  415 epsilon: 0.1641
```

```
score: 150.0
starting episode  416 epsilon: 0.1636
score: 225.0
starting episode  417 epsilon: 0.1630
score: 135.0
starting episode  418 epsilon: 0.1624
score: 250.0
starting episode  419 epsilon: 0.1619
score: 210.0
starting episode  420 epsilon: 0.1613
score: 210.0
starting episode  421 epsilon: 0.1608
score: 165.0
starting episode  422 epsilon: 0.1602
score: 195.0
starting episode  423 epsilon: 0.1597
score: 180.0
starting episode  424 epsilon: 0.1591
score: 240.0
starting episode  425 epsilon: 0.1586
score: 195.0
starting episode  426 epsilon: 0.1580
score: 225.0
starting episode  427 epsilon: 0.1575
score: 140.0
starting episode  428 epsilon: 0.1570
score: 165.0
starting episode  429 epsilon: 0.1564
score: 670.0
starting episode  430 epsilon: 0.1559
score: 665.0
starting episode  431 epsilon: 0.1554
score: 170.0
starting episode  432 epsilon: 0.1548
score: 200.0
starting episode  433 epsilon: 0.1543
score: 230.0
starting episode  434 epsilon: 0.1538
score: 250.0
starting episode  435 epsilon: 0.1533
score: 190.0
starting episode  436 epsilon: 0.1528
score: 155.0
starting episode  437 epsilon: 0.1522
score: 155.0
starting episode  438 epsilon: 0.1517
score: 305.0
starting episode  439 epsilon: 0.1512
```

```
score: 245.0
starting episode  440 epsilon: 0.1507
score: 215.0
starting episode  441 epsilon: 0.1502
score: 235.0
starting episode  442 epsilon: 0.1497
score: 250.0
starting episode  443 epsilon: 0.1492
score: 185.0
starting episode  444 epsilon: 0.1487
score: 225.0
starting episode  445 epsilon: 0.1482
score: 165.0
starting episode  446 epsilon: 0.1477
score: 225.0
starting episode  447 epsilon: 0.1473
score: 225.0
starting episode  448 epsilon: 0.1468
score: 270.0
starting episode  449 epsilon: 0.1463
score: 560.0
starting episode  450 epsilon: 0.1458
score: 530.0
starting episode  451 epsilon: 0.1453
score: 160.0
starting episode  452 epsilon: 0.1449
score: 280.0
starting episode  453 epsilon: 0.1444
score: 250.0
starting episode  454 epsilon: 0.1439
score: 250.0
starting episode  455 epsilon: 0.1434
score: 425.0
starting episode  456 epsilon: 0.1430
score: 705.0
starting episode  457 epsilon: 0.1425
score: 305.0
starting episode  458 epsilon: 0.1421
score: 250.0
starting episode  459 epsilon: 0.1416
score: 285.0
starting episode  460 epsilon: 0.1411
score: 180.0
starting episode  461 epsilon: 0.1407
score: 190.0
starting episode  462 epsilon: 0.1402
score: 215.0
starting episode  463 epsilon: 0.1398
```

```
score: 410.0
starting episode  464 epsilon: 0.1393
score: 345.0
starting episode  465 epsilon: 0.1389
score: 395.0
starting episode  466 epsilon: 0.1384
score: 190.0
starting episode  467 epsilon: 0.1380
score: 325.0
starting episode  468 epsilon: 0.1376
score: 420.0
starting episode  469 epsilon: 0.1371
score: 215.0
starting episode  470 epsilon: 0.1367
score: 425.0
starting episode  471 epsilon: 0.1363
score: 260.0
starting episode  472 epsilon: 0.1358
score: 215.0
starting episode  473 epsilon: 0.1354
score: 155.0
starting episode  474 epsilon: 0.1350
score: 275.0
starting episode  475 epsilon: 0.1346
score: 240.0
starting episode  476 epsilon: 0.1341
score: 150.0
starting episode  477 epsilon: 0.1337
score: 250.0
starting episode  478 epsilon: 0.1333
score: 420.0
starting episode  479 epsilon: 0.1329
score: 305.0
starting episode  480 epsilon: 0.1325
score: 185.0
starting episode  481 epsilon: 0.1321
score: 220.0
starting episode  482 epsilon: 0.1316
score: 220.0
starting episode  483 epsilon: 0.1312
score: 420.0
starting episode  484 epsilon: 0.1308
score: 200.0
starting episode  485 epsilon: 0.1304
score: 420.0
starting episode  486 epsilon: 0.1300
score: 180.0
starting episode  487 epsilon: 0.1296
```

```
score: 220.0
starting episode  488 epsilon: 0.1292
score: 145.0
starting episode  489 epsilon: 0.1288
score: 195.0
starting episode  490 epsilon: 0.1284
score: 190.0
starting episode  491 epsilon: 0.1281
score: 235.0
starting episode  492 epsilon: 0.1277
score: 285.0
starting episode  493 epsilon: 0.1273
score: 250.0
starting episode  494 epsilon: 0.1269
score: 405.0
starting episode  495 epsilon: 0.1265
score: 275.0
starting episode  496 epsilon: 0.1261
score: 195.0
starting episode  497 epsilon: 0.1257
score: 280.0
starting episode  498 epsilon: 0.1254
score: 200.0
starting episode  499 epsilon: 0.1250
score: 275.0
starting episode  500 epsilon: 0.1246
score: 195.0
```

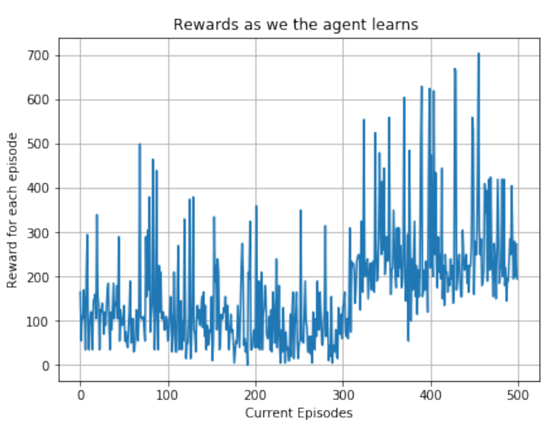
[60]:

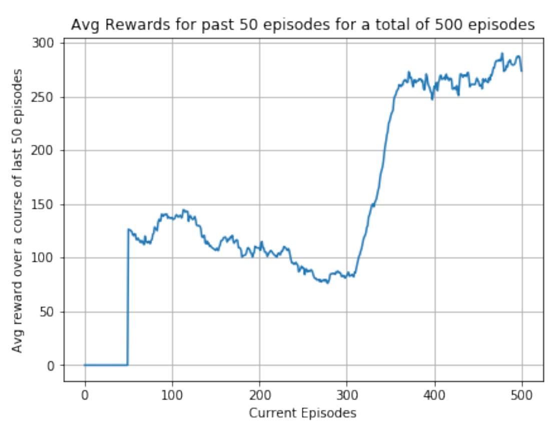
```
k = 50
avg_scores = calculate_avg_reward(scores, k)
```

Plotting graphs

[73]:

```
plot_graphs(scores, avg_scores)
```





```
[ ]:
```

```
[ ]:
```