Continuous_Control

December 29, 2020

1 Continuous Control

You are welcome to use this coding environment to train your agent for the project. Follow the instructions below to get started!

1.0.1 1. Start the Environment

Run the next code cell to install a few packages. This line will take a few minutes to run!

```
In [3]: !pip -q install ./python

        # !pip install -U pip
```

```
tensorflow 1.7.1 has requirement numpy>=1.13.3, but you'll have numpy 1.12.1 which is incompatib
ipython 6.5.0 has requirement prompt-toolkit<2.0.0,>=1.0.15, but you'll have prompt-toolkit 3.0.
```

The environments corresponding to both versions of the environment are already saved in the Workspace and can be accessed at the file paths provided below.

Please select one of the two options below for loading the environment.

```
In [4]: from unityagents import UnityEnvironment
        import numpy as np
        from collections import deque
        import matplotlib.pyplot as plt
        import numpy as np
        import random
        import time
        import torch
        from unityagents import UnityEnvironment

        %matplotlib inline
        # select this option to load version 1 (with a single agent) of the environment
        env = UnityEnvironment(file_name='/data/Reacher_One_Linux_NoVis/Reacher_One_Linux_NoVis.

        # select this option to load version 2 (with 20 agents) of the environment
        # env = UnityEnvironment(file_name='/data/Reacher_Linux_NoVis/Reacher.x86_64')
```

```
INFO:unityagents:
'Academy' started successfully!
Unity Academy name: Academy
        Number of Brains: 1
        Number of External Brains : 1
        Lesson number : 0
        Reset Parameters :
                goal_speed -> 1.0
                goal_size -> 5.0
Unity brain name: ReacherBrain
        Number of Visual Observations (per agent): 0
        Vector Observation space type: continuous
        Vector Observation space size (per agent): 33
        Number of stacked Vector Observation: 1
        Vector Action space type: continuous
        Vector Action space size (per agent): 4
        Vector Action descriptions: , , ,
```

Environments contain ***brains*** which are responsible for deciding the actions of their associated agents. Here we check for the first brain available, and set it as the default brain we will be controlling from Python.

```
In [9]: # get the default brain
        brain_name = env.brain_names[0]
        brain = env.brains[brain_name]
```

1.0.2 2. Examine the State and Action Spaces

Run the code cell below to print some information about the environment.

```
In [10]: # reset the environment
         env_info = env.reset(train_mode=True)[brain_name]

         # number of agents
         num_agents = len(env_info.agents)
         print('Number of agents:', num_agents)

         # size of each action
         action_size = brain.vector_action_space_size
         print('Size of each action:', action_size)

         # examine the state space
         states = env_info.vector_observations
         state_size = states.shape[1]
         print('There are {} agents. Each observes a state with length: {}'.format(states.shape[
         print('The state for the first agent looks like:', states[0])
```

```
Number of agents: 1
Size of each action: 4
There are 1 agents. Each observes a state with length: 33
The state for the first agent looks like: [ 0.00000000e+00 -4.00000000e+00  0.00000000e+00  1.00
 -0.00000000e+00 -0.00000000e+00 -4.37113883e-08  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00 -1.00000000e+01  0.00000000e+00
  1.00000000e+00 -0.00000000e+00 -0.00000000e+00 -4.37113883e-08
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00 -6.30408478e+00 -1.00000000e+00
 -4.92529202e+00  0.00000000e+00  1.00000000e+00  0.00000000e+00
 -5.33014059e-01]
```

1.0.3 3. Take Random Actions in the Environment

In the next code cell, you will learn how to use the Python API to control the agent and receive feedback from the environment.

Note that **in this coding environment, you will not be able to watch the agents while they are training**, and you should set `train_mode=True` to restart the environment.

```
In [8]: env_info = env.reset(train_mode=False)[brain_name]      # reset the environment
        states = env_info.vector_observations                  # get the current state (for each
        scores = np.zeros(num_agents)                          # initialize the score (for each
        while True:
            actions = np.random.randn(num_agents, action_size) # select an action (for each agen
            actions = np.clip(actions, -1, 1)                  # all actions between -1 and 1
            env_info = env.step(actions)[brain_name]           # send all actions to tne environ
            next_states = env_info.vector_observations         # get next state (for each agent)
            rewards = env_info.rewards                         # get reward (for each agent)
            dones = env_info.local_done                        # see if episode finished
            scores += env_info.rewards                         # update the score (for each agen
            states = next_states                               # roll over states to next time s
            if np.any(dones):                                  # exit loop if episode finished
                break
        print('Total score (averaged over agents) this episode: {}'.format(np.mean(scores)))


        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-8-b273097e0cc3> in <module>()
          1 env_info = env.reset(train_mode=False)[brain_name]      # reset the environment
          2 states = env_info.vector_observations                  # get the current state (for
    ----> 3 scores = np.zeros(num_agents)                          # initialize the score (for e
          4 while True:
          5     actions = np.random.randn(num_agents, action_size) # select an action (for each
```

```
NameError: name 'num_agents' is not defined
```

When finished, you can close the environment.

1.0.4 4. It's Your Turn!

Now it's your turn to train your own agent to solve the environment! A few **important notes**: - When training the environment, set `train_mode=True`, so that the line for resetting the environment looks like the following:

```
env_info = env.reset(train_mode=True)[brain_name]
```

- To structure your work, you're welcome to work directly in this Jupyter notebook, or you might like to start over with a new file! You can see the list of files in the workspace by clicking on ***Jupyter*** in the top left corner of the notebook.
- In this coding environment, you will not be able to watch the agents while they are training. However, ***after training the agents***, you can download the saved model weights to watch the agents on your own machine!

In [5]:
```
import random
import copy
from collections import deque, namedtuple

import numpy as np
import matplotlib.pyplot as plt

import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

from unityagents import UnityEnvironment

DEVICE = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")


def hidden_init(layer):
    fan_in = layer.weight.data.size()[0]
    lim = 1. / np.sqrt(fan_in)
    return -lim, lim


class Actor(nn.Module):
    def __init__(self, state_size, action_size, fc1_units, fc2_units):
        super(Actor, self).__init__()
        self.fc1 = nn.Linear(state_size, fc1_units)
```

```
        self.bn1 = nn.BatchNorm1d(fc1_units)
        self.fc2 = nn.Linear(fc1_units, fc2_units)
        self.bn2 = nn.BatchNorm1d(fc2_units)
        self.fc3 = nn.Linear(fc2_units, action_size)
        self.reset_parameters()

    def reset_parameters(self):
        self.fc1.weight.data.uniform_(*hidden_init(self.fc1))
        self.fc2.weight.data.uniform_(*hidden_init(self.fc2))
        self.fc3.weight.data.uniform_(-3e-3, 3e-3)

    def forward(self, state):
        if state.dim() == 1:
            state = torch.unsqueeze(state, 0)

        x = self.bn1(F.relu(self.fc1(state)))
        x = self.bn2(F.relu(self.fc2(x)))
        return F.tanh(self.fc3(x))


class Critic(nn.Module):
    def __init__(self, state_size, action_size, fc1_units, fc2_units):
        super(Critic, self).__init__()
        self.fc1 = nn.Linear(state_size, fc1_units)
        self.bn1 = nn.BatchNorm1d(fc1_units)
        self.fc2 = nn.Linear(fc1_units + action_size, fc2_units)
        self.fc3 = nn.Linear(fc2_units, 1)
        self.reset_parameters()

    def reset_parameters(self):
        self.fc1.weight.data.uniform_(*hidden_init(self.fc1))
        self.fc2.weight.data.uniform_(*hidden_init(self.fc2))
        self.fc3.weight.data.uniform_(-3e-3, 3e-3)

    def forward(self, state, action):
        if state.dim() == 1:
            state = torch.unsqueeze(state, 0)

        x = self.bn1(F.relu(self.fc1(state)))
        x = F.relu(self.fc2(torch.cat((x, action), dim=1)))
        return self.fc3(x)


Experience = namedtuple('Experience', 'state action reward next_state done')


class Replay:
    def __init__(self, action_size, buffer_size, batch_size):
```

```
        self.action_size = action_size
        self.buffer = deque(maxlen=buffer_size)
        self.batch_size = batch_size

    def add(self, state, action, reward, next_state, done):
        experience = Experience(state, action, reward, next_state, done)
        self.buffer.append(experience)

    def sample(self):
        experiences = random.sample(self.buffer, k=self.batch_size)

        states = torch.from_numpy(np.vstack([e.state for e in experiences])).float().to(
        actions = torch.from_numpy(np.vstack([e.action for e in experiences])).float().t
        rewards = torch.from_numpy(np.vstack([e.reward for e in experiences])).float().t
        next_states = torch.from_numpy(np.vstack([e.next_state for e in experiences])).f
        dones = torch.from_numpy(np.vstack([e.done for e in experiences]).astype(np.uint

        return states, actions, rewards, next_states, dones

    def __len__(self):
        return len(self.buffer)


class Agent:
    def __init__(self, config):
        self.config = config

        self.online_actor = config.actor_fn().to(DEVICE)
        self.target_actor = config.actor_fn().to(DEVICE)
        self.actor_opt = config.actor_opt_fn(self.online_actor.parameters())

        self.online_critic = config.critic_fn().to(DEVICE)
        self.target_critic = config.critic_fn().to(DEVICE)
        self.critic_opt = config.critic_opt_fn(self.online_critic.parameters())

        self.noise = config.noise_fn()
        self.replay = config.replay_fn()

    def step(self, state, action, reward, next_state, done):
        self.replay.add(state, action, reward, next_state, done)

        if len(self.replay) > self.replay.batch_size:
            self.learn()

    def act(self, state, add_noise=True):
        state = torch.from_numpy(state).float().to(DEVICE)

        self.online_actor.eval()
```

```
        with torch.no_grad():
            action = self.online_actor(state).cpu().data.numpy()

        self.online_actor.train()

        if add_noise:
            action += self.noise.sample()

        return np.clip(action, -1, 1)

    def reset(self):
        self.noise.reset()

    def learn(self):
        states, actions, rewards, next_states, dones = self.replay.sample()

        # Update online critic model
        # Predict actions for the next states with the target actor model
        target_next_actions = self.target_actor(next_states)
        # Compute Q values for the next states and actions with the target critic model
        target_next_qs = self.target_critic(next_states, target_next_actions)
        # Compute target Q values for the current states using the Bellman equation
        target_qs = rewards + (self.config.discount * target_next_qs * (1 - dones))
        # Compute Q values for the current states and actions with the online critic mod
        online_qs = self.online_critic(states, actions)
        # Compute and minimize the online critic loss
        critic_loss = F.mse_loss(online_qs, target_qs)
        self.critic_opt.zero_grad()
        critic_loss.backward()
        torch.nn.utils.clip_grad_norm_(self.online_critic.parameters(), 1)
        self.critic_opt.step()

        # Update online actor model
        # Predict actions for current states from the online actor model
        online_actions = self.online_actor(states)
        # Compute and minimize the online actor loss
        actor_loss = -self.online_critic(states, online_actions).mean()
        self.actor_opt.zero_grad()
        actor_loss.backward()
        self.actor_opt.step()

        # Update target critic and actor models
        self.soft_update(self.online_critic, self.target_critic)
        self.soft_update(self.online_actor, self.target_actor)

    def soft_update(self, online_model, target_model):
        for target_param, online_param in zip(target_model.parameters(), online_model.pa
```

```
            target_param.data.copy_(self.config.target_mix * online_param.data + (1.0 -


class OrnsteinUhlenbeck:
    def __init__(self, size, mu, theta, sigma):
        self.state = None
        self.mu = mu * np.ones(size)
        self.theta = theta
        self.sigma = sigma
        self.reset()

    def reset(self):
        self.state = copy.copy(self.mu)

    def sample(self):
        x = self.state
        dx = self.theta * (self.mu - x) + self.sigma * np.array([random.random() for _ i
        self.state = x + dx
        return self.state


def run(agent):
    config = agent.config
    scores_deque = deque(maxlen=100)
    scores = []

    for episode in range(1, config.max_episodes + 1):
        agent.reset()
        score = 0

        env_info = config.env.reset(train_mode=True)[config.brain_name]
        state = env_info.vector_observations[0]

        for step in range(config.max_steps):
            action = agent.act(state)
            env_info = config.env.step(action)[config.brain_name]
            next_state = env_info.vector_observations[0]
            reward = env_info.rewards[0]
            done = env_info.local_done[0]

            agent.step(state, action, reward, next_state, done)

            score += reward
            state = next_state

            if done:
                break
```

```
        scores.append(score)
        scores_deque.append(score)
        mean_score = np.mean(scores_deque)

        print('\rEpisode {}\tAverage Score: {:.2f}\tScore: {:.2f}'.format(episode, mean_

        if mean_score >= config.goal_score:
            break

    torch.save(agent.online_actor.state_dict(), config.actor_path)
    torch.save(agent.online_critic.state_dict(), config.critic_path)

    fig, ax = plt.subplots()
    ax.plot(np.arange(1, len(scores) + 1), scores)
    ax.set_ylabel('Score')
    ax.set_xlabel('Episode #')
    fig.savefig(config.scores_path)
    plt.show()


class Config:
    def __init__(self, seed):
        self.seed = seed
        random.seed(seed)
        torch.manual_seed(seed)

        self.env = None
        self.brain_name = None
        self.state_size = None
        self.action_size = None
        self.actor_fn = None
        self.actor_opt_fn = None
        self.critic_fn = None
        self.critic_opt_fn = None
        self.replay_fn = None
        self.noise_fn = None
        self.discount = None
        self.target_mix = None

        self.max_episodes = None
        self.max_steps = None

        self.actor_path = None
        self.critic_path = None
        self.scores_path = None


def main():
```

```
    config = Config(seed=6)
    config.env =UnityEnvironment(file_name='/data/Reacher_One_Linux_NoVis/Reacher_One_Li
    config.brain_name = config.env.brain_names[0]
    config.state_size = config.env.brains[config.brain_name].vector_observation_space_si
    config.action_size = config.env.brains[config.brain_name].vector_action_space_size

    config.actor_fn = lambda: Actor(config.state_size, config.action_size, fc1_units=256
    config.actor_opt_fn = lambda params: optim.Adam(params, lr=3e-4)

    config.critic_fn = lambda: Critic(config.state_size, config.action_size, fc1_units=2
    config.critic_opt_fn = lambda params: optim.Adam(params, lr=3e-4)

    config.replay_fn = lambda: Replay(config.action_size, buffer_size=int(1e6), batch_si
    config.noise_fn = lambda: OrnsteinUhlenbeck(config.action_size, mu=0., theta=0.15, s

    config.discount = 0.99
    config.target_mix = 1e-3

    config.max_episodes = int(1000)
    config.max_steps = int(1e6)
    config.goal_score = 30

    config.actor_path = 'actor.pth'
    config.critic_path = 'critic.pth'
    config.scores_path = 'scores.png'

    agent = Agent(config)
    run(agent)


if __name__ == '__main__':
    main()
```

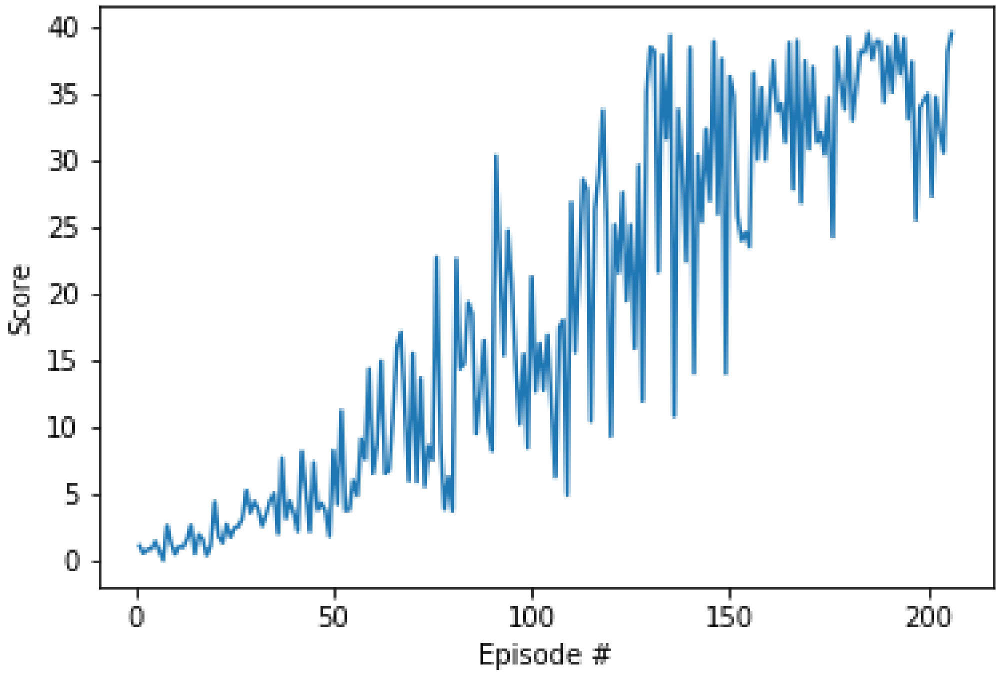
```
Episode 1       Average Score: 1.10       Score: 1.10
Episode 2       Average Score: 0.81       Score: 0.53
Episode 3       Average Score: 0.81       Score: 0.79
Episode 4       Average Score: 0.82       Score: 0.85
Episode 5       Average Score: 0.94       Score: 1.41
Episode 6       Average Score: 0.89       Score: 0.69
Episode 7       Average Score: 0.77       Score: 0.00
Episode 8       Average Score: 0.99       Score: 2.58
Episode 9       Average Score: 1.03       Score: 1.29
Episode 10       Average Score: 0.97       Score: 0.46
Episode 11       Average Score: 0.98       Score: 1.04
Episode 12       Average Score: 0.98       Score: 0.97
Episode 13       Average Score: 1.02       Score: 1.56
Episode 14       Average Score: 1.13       Score: 2.60
Episode 15       Average Score: 1.09       Score: 0.48
```

```
Episode 16      Average Score: 1.14     Score: 1.91
Episode 17      Average Score: 1.17     Score: 1.58
Episode 18      Average Score: 1.12     Score: 0.37
Episode 19      Average Score: 1.12     Score: 1.08
Episode 20      Average Score: 1.28     Score: 4.38
Episode 21      Average Score: 1.31     Score: 1.77
Episode 22      Average Score: 1.31     Score: 1.33
Episode 23      Average Score: 1.37     Score: 2.71
Episode 24      Average Score: 1.38     Score: 1.74
Episode 25      Average Score: 1.43     Score: 2.42
Episode 26      Average Score: 1.47     Score: 2.55
Episode 27      Average Score: 1.53     Score: 3.14
Episode 28      Average Score: 1.66     Score: 5.24
Episode 29      Average Score: 1.73     Score: 3.55
Episode 30      Average Score: 1.82     Score: 4.40
Episode 31      Average Score: 1.88     Score: 3.80
Episode 32      Average Score: 1.90     Score: 2.58
Episode 33      Average Score: 1.95     Score: 3.46
Episode 34      Average Score: 2.02     Score: 4.43
Episode 35      Average Score: 2.11     Score: 5.01
Episode 36      Average Score: 2.11     Score: 1.99
Episode 37      Average Score: 2.26     Score: 7.69
Episode 38      Average Score: 2.28     Score: 3.14
Episode 39      Average Score: 2.33     Score: 4.43
Episode 40      Average Score: 2.36     Score: 3.55
Episode 41      Average Score: 2.36     Score: 2.16
Episode 42      Average Score: 2.50     Score: 8.16
Episode 43      Average Score: 2.56     Score: 5.21
Episode 44      Average Score: 2.55     Score: 2.15
Episode 45      Average Score: 2.66     Score: 7.33
Episode 46      Average Score: 2.68     Score: 3.78
Episode 47      Average Score: 2.72     Score: 4.28
Episode 48      Average Score: 2.74     Score: 3.69
Episode 49      Average Score: 2.72     Score: 1.80
Episode 50      Average Score: 2.83     Score: 8.25
Episode 51      Average Score: 2.85     Score: 4.19
Episode 52      Average Score: 3.02     Score: 11.20
Episode 53      Average Score: 3.03     Score: 3.74
Episode 54      Average Score: 3.04     Score: 3.80
Episode 55      Average Score: 3.10     Score: 5.99
Episode 56      Average Score: 3.13     Score: 4.87
Episode 57      Average Score: 3.23     Score: 9.09
Episode 58      Average Score: 3.31     Score: 7.57
Episode 59      Average Score: 3.49     Score: 14.32
Episode 60      Average Score: 3.54     Score: 6.51
Episode 61      Average Score: 3.63     Score: 8.73
Episode 62      Average Score: 3.81     Score: 14.90
Episode 63      Average Score: 3.85     Score: 6.50
```

```
Episode 64        Average Score: 3.90        Score: 6.76
Episode 65        Average Score: 4.01        Score: 10.90
Episode 66        Average Score: 4.19        Score: 15.98
Episode 67        Average Score: 4.38        Score: 17.09
Episode 68        Average Score: 4.49        Score: 11.47
Episode 69        Average Score: 4.51        Score: 5.94
Episode 70        Average Score: 4.66        Score: 15.48
Episode 71        Average Score: 4.68        Score: 5.87
Episode 72        Average Score: 4.80        Score: 13.62
Episode 73        Average Score: 4.81        Score: 5.54
Episode 74        Average Score: 4.87        Score: 8.55
Episode 75        Average Score: 4.90        Score: 7.51
Episode 76        Average Score: 5.13        Score: 22.69
Episode 77        Average Score: 5.18        Score: 8.91
Episode 78        Average Score: 5.17        Score: 3.84
Episode 79        Average Score: 5.18        Score: 6.24
Episode 80        Average Score: 5.16        Score: 3.72
Episode 81        Average Score: 5.38        Score: 22.58
Episode 82        Average Score: 5.49        Score: 14.34
Episode 83        Average Score: 5.60        Score: 14.76
Episode 84        Average Score: 5.76        Score: 19.33
Episode 85        Average Score: 5.91        Score: 18.45
Episode 86        Average Score: 5.95        Score: 9.48
Episode 87        Average Score: 6.03        Score: 12.63
Episode 88        Average Score: 6.15        Score: 16.47
Episode 89        Average Score: 6.19        Score: 10.05
Episode 90        Average Score: 6.21        Score: 8.20
Episode 91        Average Score: 6.48        Score: 30.29
Episode 92        Average Score: 6.65        Score: 22.25
Episode 93        Average Score: 6.74        Score: 15.35
Episode 94        Average Score: 6.93        Score: 24.71
Episode 95        Average Score: 7.08        Score: 20.65
Episode 96        Average Score: 7.16        Score: 14.64
Episode 97        Average Score: 7.19        Score: 10.20
Episode 98        Average Score: 7.27        Score: 15.43
Episode 99        Average Score: 7.28        Score: 8.43
Episode 100        Average Score: 7.42        Score: 21.24
Episode 101        Average Score: 7.54        Score: 12.63
Episode 102        Average Score: 7.70        Score: 16.27
Episode 103        Average Score: 7.82        Score: 12.71
Episode 104        Average Score: 7.98        Score: 16.89
Episode 105        Average Score: 8.08        Score: 11.84
Episode 106        Average Score: 8.14        Score: 6.25
Episode 107        Average Score: 8.31        Score: 17.43
Episode 108        Average Score: 8.46        Score: 17.99
Episode 109        Average Score: 8.50        Score: 4.89
Episode 110        Average Score: 8.76        Score: 26.80
Episode 111        Average Score: 8.91        Score: 15.57
```

```
Episode 112        Average Score: 9.12        Score: 21.67
Episode 113        Average Score: 9.39        Score: 28.53
Episode 114        Average Score: 9.64        Score: 27.77
Episode 115        Average Score: 9.74        Score: 10.43
Episode 116        Average Score: 9.98        Score: 26.33
Episode 117        Average Score: 10.25        Score: 28.92
Episode 118        Average Score: 10.59        Score: 33.73
Episode 119        Average Score: 10.83        Score: 25.10
Episode 120        Average Score: 10.88        Score: 9.29
Episode 121        Average Score: 11.11        Score: 25.15
Episode 122        Average Score: 11.31        Score: 21.54
Episode 123        Average Score: 11.56        Score: 27.56
Episode 124        Average Score: 11.74        Score: 19.45
Episode 125        Average Score: 11.97        Score: 25.13
Episode 126        Average Score: 12.10        Score: 15.89
Episode 127        Average Score: 12.36        Score: 29.58
Episode 128        Average Score: 12.43        Score: 11.92
Episode 129        Average Score: 12.75        Score: 35.09
Episode 130        Average Score: 13.09        Score: 38.42
Episode 131        Average Score: 13.43        Score: 38.15
Episode 132        Average Score: 13.62        Score: 21.60
Episode 133        Average Score: 13.96        Score: 37.84
Episode 134        Average Score: 14.24        Score: 31.60
Episode 135        Average Score: 14.58        Score: 39.32
Episode 136        Average Score: 14.67        Score: 10.78
Episode 137        Average Score: 14.93        Score: 33.79
Episode 138        Average Score: 15.19        Score: 29.43
Episode 139        Average Score: 15.37        Score: 22.40
Episode 140        Average Score: 15.72        Score: 38.39
Episode 141        Average Score: 15.84        Score: 14.02
Episode 142        Average Score: 16.06        Score: 30.33
Episode 143        Average Score: 16.26        Score: 25.37
Episode 144        Average Score: 16.56        Score: 32.30
Episode 145        Average Score: 16.76        Score: 26.91
Episode 146        Average Score: 17.11        Score: 38.87
Episode 147        Average Score: 17.33        Score: 25.92
Episode 148        Average Score: 17.66        Score: 37.55
Episode 149        Average Score: 17.79        Score: 13.98
Episode 150        Average Score: 18.07        Score: 36.24
Episode 151        Average Score: 18.37        Score: 34.81
Episode 152        Average Score: 18.52        Score: 25.78
Episode 153        Average Score: 18.72        Score: 23.93
Episode 154        Average Score: 18.93        Score: 24.57
Episode 155        Average Score: 19.10        Score: 23.47
Episode 156        Average Score: 19.42        Score: 36.52
Episode 157        Average Score: 19.63        Score: 30.01
Episode 158        Average Score: 19.91        Score: 35.42
Episode 159        Average Score: 20.06        Score: 29.98
```

```
Episode 160        Average Score: 20.34        Score: 33.99
Episode 161        Average Score: 20.62        Score: 37.41
Episode 162        Average Score: 20.81        Score: 33.62
Episode 163        Average Score: 21.09        Score: 34.25
Episode 164        Average Score: 21.33        Score: 31.33
Episode 165        Average Score: 21.61        Score: 38.79
Episode 166        Average Score: 21.73        Score: 27.82
Episode 167        Average Score: 21.95        Score: 38.94
Episode 168        Average Score: 22.10        Score: 26.79
Episode 169        Average Score: 22.42        Score: 37.41
Episode 170        Average Score: 22.57        Score: 30.83
Episode 171        Average Score: 22.88        Score: 36.97
Episode 172        Average Score: 23.06        Score: 31.34
Episode 173        Average Score: 23.33        Score: 32.06
Episode 174        Average Score: 23.54        Score: 30.39
Episode 175        Average Score: 23.82        Score: 34.68
Episode 176        Average Score: 23.83        Score: 24.25
Episode 177        Average Score: 24.13        Score: 38.40
Episode 178        Average Score: 24.45        Score: 35.88
Episode 179        Average Score: 24.72        Score: 33.79
Episode 180        Average Score: 25.08        Score: 39.16
Episode 181        Average Score: 25.18        Score: 32.94
Episode 182        Average Score: 25.39        Score: 35.48
Episode 183        Average Score: 25.63        Score: 38.15
Episode 184        Average Score: 25.81        Score: 38.01
Episode 185        Average Score: 26.02        Score: 39.50
Episode 186        Average Score: 26.30        Score: 37.53
Episode 187        Average Score: 26.57        Score: 38.89
Episode 188        Average Score: 26.79        Score: 38.85
Episode 189        Average Score: 27.03        Score: 34.33
Episode 190        Average Score: 27.33        Score: 38.47
Episode 191        Average Score: 27.38        Score: 35.09
Episode 192        Average Score: 27.55        Score: 39.33
Episode 193        Average Score: 27.76        Score: 36.42
Episode 194        Average Score: 27.91        Score: 39.09
Episode 195        Average Score: 28.03        Score: 33.06
Episode 196        Average Score: 28.26        Score: 37.34
Episode 197        Average Score: 28.41        Score: 25.51
Episode 198        Average Score: 28.60        Score: 33.87
Episode 199        Average Score: 28.86        Score: 34.46
Episode 200        Average Score: 28.99        Score: 34.94
Episode 201        Average Score: 29.14        Score: 27.29
Episode 202        Average Score: 29.32        Score: 34.69
Episode 203        Average Score: 29.52        Score: 32.13
Episode 204        Average Score: 29.66        Score: 30.54
Episode 205        Average Score: 29.92        Score: 38.15
Episode 206        Average Score: 30.25        Score: 39.49
```


Score
Episode #
0
5
10
15
20
25
30
35
40
0
50
100
150
200