03_lunar_lander_deep_q_learning

September 29, 2021

1 Double Deep Q-Learning

1.1 The Open AI Lunar Lander environment

The OpenAI Gym is a RL platform that provides standardized environments to test and benchmark RL algorithms using Python. It is also possible to extend the platform and register custom environments.

The Lunar Lander (LL) environment requires the agent to control its motion in two dimensions, based on a discrete action space and low-dimensional state observations that include position, orientation, and velocity. At each time step, the environment provides an observation of the new state and a positive or negative reward. Each episode consists of up to 1,000 time steps. The following diagram shows selected frames from a successful landing after 250 episodes by the agent we will train:

More specifically, the agent observes eight aspects of the state, including six continuous and two discrete elements. Based on the observed elements, the agent knows its location, direction, speed of movement, and whether it has (partially) landed. However, it does not know where it should be moving using its available actions or observe the inner state of the environment in the sense of understanding the rules that govern its motion.

At each time step, the agent controls its motion using one of four discrete actions. It can do nothing (and continue on its current path), fire its main engine (to reduce downward motion), or steer to the left or right using the respective orientation engines. There are no fuel limitations.

The goal is to land the agent between two flags on a landing pad at coordinates (0, 0), but landing outside of the pad is possible. The agent accumulates rewards in the range of 100-140 for moving toward the pad, depending on the exact landing spot. However, moving away from the target negates the reward the agent would have gained by moving toward the pad. Ground contact by each leg adds ten points, and using the main engine costs -0.3 points.

An episode terminates if the agent lands or crashes, adding or subtracting 100 points, respectively, or after 1,000 time steps. Solving LL requires achieving a cumulative reward of at least 200 on average over 100 consecutive episodes.

1.2 Deep Q-Learning

Deep Q learning estimates the value of the available actions for a given state using a deep neural network. It was introduced by Deep Mind's Playing Atari with Deep Reinforcement Learning (2013), where RL agents learned to play games solely from pixel input.

The Deep Q-Learning algorithm approximates the action-value function q by learning a set of weights of a multi-layered Deep Q Network (DQN) that maps states to actions so that

$$q(s, a, \theta) \approx q^*(s, a)$$

The algorithm applies gradient descent to a loss function defined as the squared difference between the DQN's estimate of the target

$$y_i = \mathbb{E}[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1} \mid s, a)]$$

and its estimate of the action-value of the current state-action pair to learn the network parameters:

$$L_i(\theta_i) = \mathbb{E}\left[\left(\overbrace{\underbrace{y_i}_{\text{Q Target}} - \underbrace{Q(s, a; \theta)}_{\text{Current Prediction}}}^{\text{TD Error}}\right)^2\right]$$

Both the target and the current estimate depend on the set of weights, underlining the distinction from supervised learning where targets are fixed prior to training.

1.2.1 Extensions

Several innovations have improved the accuracy and convergence speed of deep Q-Learning, namely: - **Experience replay** stores a history of state, action, reward, and next state transitions and randomly samples mini-batches from this experience to update the network weights at each time step before the agent selects an -greedy action. It increases sample efficiency, reduces the autocorrelation of samples, and limits the feedback due to the current weights producing training samples that can lead to local minima or divergence. - **Slowly-changing target network** weakens the feedback loop from the current network parameters on the neural network weight updates. Also invented by by Deep Mind in Human-level control through deep reinforcement learning (2015), it use a slowly-changing target network that has the same architecture as the Q-network, but its weights are only updated periodically. The target network generates the predictions of the next state value used to update the Q-Networks estimate of the current state's value. - **Double deep Q-learning** addresses the bias of deep Q-Learning to overestimate action values because it purposely samples the highest action value. This bias can negatively affect the learning process and the resulting policy if it does not apply uniformly , as shown by Hado van Hasselt in Deep Reinforcement Learning with Double Q-learning (2015). To decouple the estimation of action values from the selection of actions, Double Deep Q-Learning (DDQN) uses the weights, of one network to select the best action given the next state, and the weights of another network to provide the corresponding action value estimate.

1.3 Imports & Settings

[1]:
```
%matplotlib inline
from pathlib import Path
from collections import deque, namedtuple
from time import time
from random import sample
import numpy as np
```

```
from numpy.random import random, randint, seed
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

import tensorflow as tf

import gym
from gym import wrappers
```

[2]:
```
sns.set_style('darkgrid', {'axes.grid' : False})
```

1.3.1 Result display helper functions

[3]:
```
def format_time(t):
    m_, s = divmod(t, 60)
    h, m = divmod(m_, 60)
    return '{:02.0f}:{:05.2f}'.format(m, s)
```

[4]:
```
def track_results(episode, episode_reward,
                  rewards_ma, rewards_std,
                  episode_steps, episode_time,
                  epsilon):
    time_ma = np.mean([episode_time[-100:]])
    T = np.sum(episode_time)

    print('{:>4d} | Reward: {:>5.0f} | MA: {:>5.0f} | '
          'Steps: {:>4d} Time: {:>5.2f} | MA: {:>5.2f} | Total: {} | '
          'eps: {:>6.3f}'.format(episode,
                                 episode_reward,
                                 rewards_ma,
                                 episode_steps[-1],
                                 episode_time[-1],
                                 time_ma,
                                 format_time(T),
                                 epsilon))
```

1.3.2 Enable virtual display to run from docker container

This is only required if you run this on server that does not have a display.

[4]:
```
from pyvirtualdisplay import Display
virtual_display = Display(visible=0, size=(1400, 900))
virtual_display.start()
```

[4]:
```
<Display cmd_param=['Xvfb', '-br', '-nolisten', 'tcp', '-screen', '0',
'1400x900x24', ':1017'] cmd=['Xvfb', '-br', '-nolisten', 'tcp', '-screen', '0',
'1400x900x24', ':1017'] oserror=None return_code=None stdout="None"
stderr="None" timeout_happened=False>
```

1.4 Set up Gym Environment

We will begin by instantiating and extracting key parameters from the LL environment:

[5]:
```
env = gym.make('LunarLander-v2')
state_dim = env.observation_space.shape[0]  # number of dimensions in state
n_actions = env.action_space.n  # number of actions
max_episode_steps = env.spec.max_episode_steps  # max number of steps per␣
↪episode
env.seed(42)
```

[5]:
```
[42]
```

1.4.1 Define hyperparameters

The agent's performance is quite sensitive to several hyperparameters. We will start with the discount and learning rates:

[ ]:
```
gamma=.99,  # discount factor
learning_rate=5e-5  # learning rate
```

We will update the target network every 100 time steps, store up to 1 m past episodes in the replay memory, and sample mini-batches of 1,024 from memory to train the agent:

[14]:
```
tau=100  # target network update frequency
replay_capacity=int(1e6)
minibatch_size=1024
```

The -greedy policy starts with pure exploration at =1, linear decay to 0.05 over 20,000 time steps, and exponential decay thereafter:

[ ]:
```
epsilon_start=1.0
epsilon_end=0.05
epsilon_linear_steps=2e4
epsilon_exp_decay=0.99
```

[6]:
```
layers=(256,) * 3  # units per layer

l2_reg=1e-6  # L2 regularization



video_freq=50
```

1.5 Create Neural Network

We will use TensorFlow to create our Double Deep Q-Network .

[ ]:
```
tf.reset_default_graph()
```

1.5.1 Dense Layers

The `create_network` function generates the three dense layers that can be trained and/or reused as required by the Q network and its slower-moving target network:

[7]:
```
def create_network(s, layers, trainable, reuse, n_actions=4):
    """Generate Q and target network with same structure"""
    for layer, units in enumerate(layers):
        x = tf.layers.dense(inputs=s if layer == 0 else x,
                            units=units,
                            activation=tf.nn.relu,
                            trainable=trainable,
                            reuse=reuse,
                            name='dense_{}'.format(layer))
    return tf.squeeze(tf.layers.dense(inputs=x,
                                      units=n_actions,
                                      trainable=trainable,
                                      reuse=reuse,
                                      name='output'))
```

1.5.2 Placeholders

Key elements of the DDQN's computational graph include placeholder variables for the state, action, and reward sequences:

[8]:
```
state = tf.placeholder(dtype=tf.float32, shape=[None, state_dim])  # input to Q␣
↪network
next_state = tf.placeholder(dtype=tf.float32, shape=[None, state_dim])  # input␣
↪to target network
action = tf.placeholder(dtype=tf.int32, shape=[None])  # action indices␣
↪(indices of Q network output)
reward = tf.placeholder(dtype=tf.float32, shape=[None])  # rewards for target␣
↪computation
not_done = tf.placeholder(dtype=tf.float32, shape=[None])  # indicators for␣
↪target computation
```

1.5.3 Episode Counter

We add a variable to keep track of episodes:

[ ]:
```
episode_count = tf.Variable(0.0, trainable=False, name='episode_count')
add_episode = episode_count.assign_add(1)
```

1.5.4 Deep Q Networks

We will create two DQNs to predict q values for the current and next state, where we hold the weights for the second network that's fixed when predicting the next state:

[ ]:
```
with tf.variable_scope('Q_Network'):
    # Q network applied to current observation
    q_action_values = create_network(state,
                                     layers=layers,
                                     trainable=True,
                                     reuse=False)

    # Q network applied to next_observation
    next_q_action_values = tf.stop_gradient(create_network(next_state,
                                                           layers=layers,
                                                           trainable=False,
                                                           reuse=True))
```

1.5.5 Slow-Moving Target Network

In addition, we will create the target network that we update every tau periods:

[ ]:
```
with tf.variable_scope('Target_Network', reuse=False):
    target_action_values = tf.stop_gradient(create_network(next_state,
                                                           layers=layers,
                                                           trainable=False,
                                                           reuse=False))
```

1.5.6 Collect Variables and Operations

To build TensorFlow's computational graph, we need to collect the relevant variables and operations:

[ ]:
```
q_network_variables = tf.get_collection(tf.GraphKeys.TRAINABLE_VARIABLES,␣
 ↪scope='Q_Network')
target_network_variables = tf.get_collection(tf.GraphKeys.GLOBAL_VARIABLES,␣
 ↪scope='Target_Network')

# update target network weights
update_target_ops = []
for i, target_variable in enumerate(target_network_variables):
    update_target_op = target_variable.assign(q_network_variables[i])
    update_target_ops.append(update_target_op)
update_target_op = tf.group(*update_target_ops, name='update_target')
```

1.5.7 Compute Q-Learning updates

The target, yi, and the predicted q value is computed as follows:

[ ]:
```
# Q target calculation
targets = reward + not_done * gamma * tf.gather_nd(target_action_values, tf.
 ↪stack(
                (tf.range(minibatch_size), tf.cast(tf.
 ↪argmax(next_q_action_values, axis=1), tf.int32)), axis=1))


# Estimated Q values for (s,a) from experience replay
predicted_q_value = tf.gather_nd(q_action_values,
                                 tf.stack((tf.range(minibatch_size),
                                           action), axis=1))
```

1.5.8 Compute Loss Function

Finally, the TD loss function that's used for stochastic gradient descent is the mean squared error (MSE) between the target and prediction:

[ ]:
```
losses = tf.squared_difference(targets, predicted_q_value)
loss = tf.reduce_mean(losses)
loss += tf.add_n([tf.nn.l2_loss(var) for var in q_network_variables if 'bias'␣
 ↪not in var.name]) * l2_reg * 0.5
```

1.5.9 Tensorboard summaries

To view results in tensorboard, we need to define summaries:

[ ]:
```
summaries = tf.summary.merge([
    tf.summary.scalar('episode', episode_count),
    tf.summary.scalar('loss', loss),
    tf.summary.scalar('max_q_value', tf.reduce_max(predicted_q_value)),
    tf.summary.histogram('loss_hist', losses),
    tf.summary.histogram('q_values', predicted_q_value)])
```

1.5.10 Set optimizer

We'll use the AdamOptimizer:

[ ]:
```
train_op = tf.train.AdamOptimizer(learning_rate).minimize(loss,
                                                          global_step=tf.train.
 ↪create_global_step())
```

1.5.11 Initialize TensorFlow session

[9]:
```
sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

```
WARNING:tensorflow:From
/home/stefan/.pyenv/versions/miniconda3-latest/envs/ml4t/lib/python3.6/site-
```

```
packages/tensorflow/python/framework/op_def_library.py:263: colocate_with (from
tensorflow.python.framework.ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From <ipython-input-7-178a7f73e60f>:9: dense (from
tensorflow.python.layers.core) is deprecated and will be removed in a future
version.
Instructions for updating:
Use keras.layers.dense instead.
WARNING:tensorflow:From
/home/stefan/.pyenv/versions/miniconda3-latest/envs/ml4t/lib/python3.6/site-
packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from
tensorflow.python.ops.math_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.cast instead.
```

1.6 Run Experiment

1.6.1 Set parameters

[10]:
```
experiment = 1
train = True
total_steps, max_episodes = 0, 1000
```

1.6.2 Initialize variables

[16]:
```
experience = deque(maxlen=replay_capacity)
episode_time, episode_steps, episode_rewards, episode_eps = [], [], [], []
```

[14]:
```
epsilon = epsilon_start
epsilon_linear_step = (epsilon_start - epsilon_end) / epsilon_linear_steps
```

1.6.3 Set up directories

[17]:
```
experiment_dir = Path('results')
monitor_path = experiment_dir / 'monitor'
checkpoint_dir = experiment_dir / 'checkpoints'
checkpoint_path = checkpoint_dir / 'model'

for path in [checkpoint_path, monitor_path]:
    if not path.exists():
        path.mkdir(exist_ok=True, parents=True)
```

1.6.4 Set up Tensorflow Logging

[12]:
```
log_path = experiment_dir / 'tensorboard'
summary_writer = tf.summary.FileWriter(logdir=log_path / 'experiment_{}'.
 ↪format(experiment))

saver = tf.train.Saver()
if not train:
    latest_checkpoint = tf.train.latest_checkpoint(checkpoint_dir.as_posix())
    if latest_checkpoint:
        saver.restore(sess, latest_checkpoint)
```

1.6.5 Set up Gym Monitoring

We will also use the built-in wrappers that permit the periodic storing of videos that display the agent's performance:

[ ]:
```
env = wrappers.Monitor(env,
                       directory=monitor_path.as_posix(),
                       video_callable=lambda count: count % video_freq == 0,
                      force=True)
```

1.6.6 Run Training Loop

[15]:
```
for episode in range(max_episodes):
    episode_start = time()
    episode_reward = 0
    episode_eps.append(epsilon)

    # Initial state
    this_observation = env.reset()
    for episode_step in range(max_episode_steps):

        # choose action according to epsilon-greedy policy wrt Q
        if train:
            if random() < epsilon:
                selected_action = randint(n_actions)
            else:
                q_s = sess.run(q_action_values,
                               feed_dict={state: this_observation[None]})
                selected_action = np.argmax(q_s)
        else:
            q_s = sess.run(q_action_values,
                           feed_dict={state: this_observation[None]})
            selected_action = np.argmax(q_s)

        next_observation, step_reward, done, _ = env.step(selected_action)
```

```
        episode_reward += step_reward

        if train:
            # add to replay buffer
            experience.append((this_observation,
                               selected_action,
                               step_reward,
                               next_observation,
                               0.0 if done else 1.0))

            # update the target weights
            if total_steps % tau == 0:
                _ = sess.run(update_target_op)

            # update weights using minibatch of (s,a,r,s') samples from␣
↪experience
            if len(experience) >= minibatch_size:
                minibatch = map(np.array, zip(*sample(experience,␣
↪minibatch_size)))
                states_batch, action_batch, reward_batch, next_states_batch,␣
↪done_batch = minibatch

                # do a train_op with required inputs
                feed_dict = {
                    state     : states_batch,
                    action    : action_batch,
                    reward    : reward_batch,
                    next_state: next_states_batch,
                    not_done  : done_batch}
                if total_steps % 100 == 0:
                    summary, global_step, _ = sess.run([summaries, tf.train.
↪get_global_step(), train_op],
                                                       feed_dict=feed_dict)
                    summary_writer.add_summary(summary, global_step)
                else:
                    _ = sess.run([train_op],
                                 feed_dict=feed_dict)

        this_observation = next_observation
        total_steps += 1

        # linearly decay epsilon from epsilon_start to epsilon_end for␣
↪epsilon_linear_steps
        if total_steps < epsilon_linear_steps:
            epsilon -= epsilon_linear_step
        # then exponentially decay every episode
        elif done:
```

```
            epsilon *= epsilon_exp_decay

        if done:
            # Increment episode counter
            episode_, _ = sess.run([episode_count, add_episode])
            if train:
                episode_summary = tf.Summary()
                episode_summary.value.add(simple_value=episode_reward,
                                          node_name='episode_reward',
                                          tag='episode_reward')
                episode_summary.value.add(simple_value=np.
↪mean(episode_rewards[-100:]),
                                          node_name='episode_reward_mavg',
                                          tag='episode_reward_mavg')
                episode_summary.value.add(simple_value=episode_step,
                                          node_name='episode_steps',
                                          tag='episode_steps')
                episode_summary.value.add(simple_value=time() - episode_start,
                                          node_name='episode_duration',
                                          tag='episode_duration')
                summary_writer.add_summary(episode_summary, episode_)
                summary_writer.flush()
            break

    episode_time.append(time() - episode_start)
    episode_steps.append(episode_step)
    episode_rewards.append(episode_reward)
    rewards_ma = np.mean([episode_rewards[-100:]])
    rewards_std = np.std([episode_rewards[-100:]])

    track_results(episode, episode_reward,
                  rewards_ma, rewards_std,
                  episode_steps, episode_time,
                  epsilon)

    if rewards_ma > 200:
        saver.save(sess, checkpoint_path.as_posix())
        break

env.close()
```

```
/usr/local/lib/python3.5/dist-packages/numpy/core/fromnumeric.py:3118:
RuntimeWarning: Mean of empty slice.
  out=out, **kwargs)
/usr/local/lib/python3.5/dist-packages/numpy/core/_methods.py:85:
RuntimeWarning: invalid value encountered in double_scalars
  ret = ret.dtype.type(ret / rcount)
```

```
   0 | Reward:  -192 | MA:  -192 | Steps:  106 Time:  1.28 | MA:  1.28 | Total:
00:01.28 | eps:  0.995
   1 | Reward:  -194 | MA:  -193 | Steps:   92 Time:  0.03 | MA:  0.65 | Total:
00:01.31 | eps:  0.991
   2 | Reward:  -163 | MA:  -183 | Steps:  121 Time:  0.13 | MA:  0.48 | Total:
00:01.44 | eps:  0.985
   3 | Reward:  -126 | MA:  -169 | Steps:   92 Time:  0.01 | MA:  0.36 | Total:
00:01.45 | eps:  0.980
   4 | Reward:  -176 | MA:  -170 | Steps:  100 Time:  0.01 | MA:  0.29 | Total:
00:01.46 | eps:  0.975
   5 | Reward:  -208 | MA:  -176 | Steps:   87 Time:  0.01 | MA:  0.25 | Total:
00:01.47 | eps:  0.971
   6 | Reward:  -295 | MA:  -193 | Steps:  119 Time:  0.02 | MA:  0.21 | Total:
00:01.49 | eps:  0.966
   7 | Reward:  -140 | MA:  -187 | Steps:   94 Time:  0.01 | MA:  0.19 | Total:
00:01.50 | eps:  0.961
   8 | Reward:  -112 | MA:  -178 | Steps:   70 Time:  0.01 | MA:  0.17 | Total:
00:01.51 | eps:  0.958
   9 | Reward:  -395 | MA:  -200 | Steps:  122 Time:  0.02 | MA:  0.15 | Total:
00:01.53 | eps:  0.952
  10 | Reward:  -116 | MA:  -192 | Steps:   63 Time:  0.25 | MA:  0.16 | Total:
00:01.78 | eps:  0.949
  11 | Reward:   -63 | MA:  -181 | Steps:   73 Time:  0.26 | MA:  0.17 | Total:
00:02.04 | eps:  0.945
  12 | Reward:  -256 | MA:  -187 | Steps:  142 Time:  0.39 | MA:  0.19 | Total:
00:02.43 | eps:  0.939
  13 | Reward:  -372 | MA:  -200 | Steps:   89 Time:  0.24 | MA:  0.19 | Total:
00:02.67 | eps:  0.934
  14 | Reward:  -163 | MA:  -198 | Steps:   87 Time:  0.29 | MA:  0.20 | Total:
00:02.96 | eps:  0.930
  15 | Reward:  -224 | MA:  -200 | Steps:  104 Time:  0.29 | MA:  0.20 | Total:
00:03.25 | eps:  0.925
  16 | Reward:  -195 | MA:  -199 | Steps:  130 Time:  0.37 | MA:  0.21 | Total:
00:03.62 | eps:  0.919
  17 | Reward:  -239 | MA:  -202 | Steps:  101 Time:  0.28 | MA:  0.22 | Total:
00:03.90 | eps:  0.914
  18 | Reward:  -197 | MA:  -201 | Steps:   82 Time:  0.27 | MA:  0.22 | Total:
00:04.17 | eps:  0.910
  19 | Reward:  -283 | MA:  -205 | Steps:   89 Time:  0.25 | MA:  0.22 | Total:
00:04.42 | eps:  0.906
  20 | Reward:  -137 | MA:  -202 | Steps:   84 Time:  0.23 | MA:  0.22 | Total:
00:04.65 | eps:  0.902
  21 | Reward:   -71 | MA:  -196 | Steps:  135 Time:  0.37 | MA:  0.23 | Total:
00:05.03 | eps:  0.895
  22 | Reward:  -137 | MA:  -194 | Steps:   94 Time:  0.31 | MA:  0.23 | Total:
00:05.34 | eps:  0.891
  23 | Reward:    20 | MA:  -185 | Steps:  121 Time:  0.37 | MA:  0.24 | Total:
00:05.71 | eps:  0.885
```

```
  24 | Reward:  -108 | MA:  -182 | Steps:   75 Time:  0.25 | MA:  0.24 | Total:
00:05.96 | eps:  0.881
  25 | Reward:  -325 | MA:  -187 | Steps:   72 Time:  0.20 | MA:  0.24 | Total:
00:06.16 | eps:  0.878
  26 | Reward:   -45 | MA:  -182 | Steps:   68 Time:  0.19 | MA:  0.24 | Total:
00:06.35 | eps:  0.875
  27 | Reward:  -305 | MA:  -186 | Steps:  105 Time:  0.34 | MA:  0.24 | Total:
00:06.69 | eps:  0.870
  28 | Reward:  -170 | MA:  -186 | Steps:  123 Time:  0.35 | MA:  0.24 | Total:
00:07.04 | eps:  0.864
  29 | Reward:  -137 | MA:  -184 | Steps:   78 Time:  0.22 | MA:  0.24 | Total:
00:07.26 | eps:  0.860
  30 | Reward:  -107 | MA:  -182 | Steps:  111 Time:  0.36 | MA:  0.25 | Total:
00:07.62 | eps:  0.855
  31 | Reward:  -312 | MA:  -186 | Steps:   92 Time:  0.26 | MA:  0.25 | Total:
00:07.88 | eps:  0.850
  32 | Reward:  -121 | MA:  -184 | Steps:   77 Time:  0.22 | MA:  0.25 | Total:
00:08.10 | eps:  0.847
  33 | Reward:  -147 | MA:  -183 | Steps:   75 Time:  0.22 | MA:  0.24 | Total:
00:08.32 | eps:  0.843
  34 | Reward:   -25 | MA:  -178 | Steps:   77 Time:  0.22 | MA:  0.24 | Total:
00:08.54 | eps:  0.839
  35 | Reward:   -83 | MA:  -175 | Steps:   87 Time:  0.30 | MA:  0.25 | Total:
00:08.84 | eps:  0.835
  36 | Reward:  -138 | MA:  -174 | Steps:   86 Time:  0.25 | MA:  0.25 | Total:
00:09.09 | eps:  0.831
  37 | Reward:  -135 | MA:  -173 | Steps:  107 Time:  0.30 | MA:  0.25 | Total:
00:09.39 | eps:  0.826
  38 | Reward:  -222 | MA:  -175 | Steps:  112 Time:  0.32 | MA:  0.25 | Total:
00:09.71 | eps:  0.820
  39 | Reward:  -136 | MA:  -174 | Steps:  108 Time:  0.35 | MA:  0.25 | Total:
00:10.06 | eps:  0.815
  40 | Reward:  -109 | MA:  -172 | Steps:   64 Time:  0.18 | MA:  0.25 | Total:
00:10.24 | eps:  0.812
  41 | Reward:  -113 | MA:  -171 | Steps:   78 Time:  0.22 | MA:  0.25 | Total:
00:10.46 | eps:  0.808
  42 | Reward:  -151 | MA:  -170 | Steps:  112 Time:  0.33 | MA:  0.25 | Total:
00:10.79 | eps:  0.803
  43 | Reward:   -62 | MA:  -168 | Steps:  100 Time:  0.35 | MA:  0.25 | Total:
00:11.14 | eps:  0.798
  44 | Reward:  -186 | MA:  -168 | Steps:  108 Time:  0.33 | MA:  0.25 | Total:
00:11.47 | eps:  0.793
  45 | Reward:   -83 | MA:  -166 | Steps:  115 Time:  0.35 | MA:  0.26 | Total:
00:11.83 | eps:  0.788
  46 | Reward:  -178 | MA:  -167 | Steps:  118 Time:  0.36 | MA:  0.26 | Total:
00:12.19 | eps:  0.782
  47 | Reward:  -103 | MA:  -165 | Steps:  111 Time:  0.38 | MA:  0.26 | Total:
00:12.57 | eps:  0.777
```

```
  48 | Reward:   -62 | MA:  -163 | Steps:   78 Time:  0.24 | MA:  0.26 | Total:
00:12.81 | eps:  0.773
  49 | Reward:    -6 | MA:  -160 | Steps:  159 Time:  0.49 | MA:  0.27 | Total:
00:13.30 | eps:  0.765
  50 | Reward:   -44 | MA:  -158 | Steps:  109 Time:  1.19 | MA:  0.28 | Total:
00:14.49 | eps:  0.760
  51 | Reward:   -44 | MA:  -156 | Steps:   79 Time:  0.27 | MA:  0.28 | Total:
00:14.76 | eps:  0.756
  52 | Reward:   -64 | MA:  -154 | Steps:   60 Time:  0.18 | MA:  0.28 | Total:
00:14.94 | eps:  0.753
  53 | Reward:   -62 | MA:  -152 | Steps:  110 Time:  0.34 | MA:  0.28 | Total:
00:15.28 | eps:  0.748
  54 | Reward:   -89 | MA:  -151 | Steps:  135 Time:  0.46 | MA:  0.29 | Total:
00:15.74 | eps:  0.742
  55 | Reward:  -160 | MA:  -151 | Steps:   90 Time:  0.28 | MA:  0.29 | Total:
00:16.01 | eps:  0.737
  56 | Reward:   -72 | MA:  -150 | Steps:  140 Time:  0.43 | MA:  0.29 | Total:
00:16.44 | eps:  0.731
  57 | Reward:   -88 | MA:  -149 | Steps:   88 Time:  0.27 | MA:  0.29 | Total:
00:16.71 | eps:  0.726
  58 | Reward:   -41 | MA:  -147 | Steps:   94 Time:  0.34 | MA:  0.29 | Total:
00:17.05 | eps:  0.722
  59 | Reward:   -55 | MA:  -145 | Steps:   70 Time:  0.22 | MA:  0.29 | Total:
00:17.27 | eps:  0.718
  60 | Reward:   -85 | MA:  -144 | Steps:  110 Time:  0.34 | MA:  0.29 | Total:
00:17.61 | eps:  0.713
  61 | Reward:   -54 | MA:  -143 | Steps:   81 Time:  0.25 | MA:  0.29 | Total:
00:17.86 | eps:  0.709
  62 | Reward:    16 | MA:  -140 | Steps:  158 Time:  0.54 | MA:  0.29 | Total:
00:18.40 | eps:  0.702
  63 | Reward:  -113 | MA:  -140 | Steps:   92 Time:  0.28 | MA:  0.29 | Total:
00:18.68 | eps:  0.697
  64 | Reward:  -114 | MA:  -140 | Steps:  102 Time:  0.32 | MA:  0.29 | Total:
00:19.00 | eps:  0.692
  65 | Reward:  -216 | MA:  -141 | Steps:  101 Time:  0.32 | MA:  0.29 | Total:
00:19.32 | eps:  0.688
  66 | Reward:   -42 | MA:  -139 | Steps:   87 Time:  0.32 | MA:  0.29 | Total:
00:19.64 | eps:  0.683
  67 | Reward:   -55 | MA:  -138 | Steps:   92 Time:  0.28 | MA:  0.29 | Total:
00:19.92 | eps:  0.679
  68 | Reward:  -196 | MA:  -139 | Steps:  123 Time:  0.38 | MA:  0.29 | Total:
00:20.30 | eps:  0.673
  69 | Reward:   -24 | MA:  -137 | Steps:  142 Time:  0.49 | MA:  0.30 | Total:
00:20.79 | eps:  0.666
  70 | Reward:   -31 | MA:  -136 | Steps:  125 Time:  0.39 | MA:  0.30 | Total:
00:21.17 | eps:  0.660
  71 | Reward:    21 | MA:  -134 | Steps:  169 Time:  0.52 | MA:  0.30 | Total:
00:21.70 | eps:  0.652
```

```
  72 | Reward:   -71 | MA:  -133 | Steps:   99 Time:  0.36 | MA:  0.30 | Total:
00:22.06 | eps:  0.647
  73 | Reward:   -24 | MA:  -131 | Steps:   66 Time:  0.20 | MA:  0.30 | Total:
00:22.26 | eps:  0.644
  74 | Reward:   -15 | MA:  -130 | Steps:  144 Time:  0.45 | MA:  0.30 | Total:
00:22.71 | eps:  0.637
  75 | Reward:   -12 | MA:  -128 | Steps:  144 Time:  0.45 | MA:  0.30 | Total:
00:23.15 | eps:  0.630
  76 | Reward:  -117 | MA:  -128 | Steps:  167 Time:  0.58 | MA:  0.31 | Total:
00:23.73 | eps:  0.623
  77 | Reward:   -31 | MA:  -127 | Steps:   99 Time:  0.31 | MA:  0.31 | Total:
00:24.04 | eps:  0.618
  78 | Reward:  -191 | MA:  -128 | Steps:  203 Time:  0.69 | MA:  0.31 | Total:
00:24.73 | eps:  0.608
  79 | Reward:   -44 | MA:  -126 | Steps:  120 Time:  0.38 | MA:  0.31 | Total:
00:25.11 | eps:  0.602
  80 | Reward:   -75 | MA:  -126 | Steps:  138 Time:  0.45 | MA:  0.32 | Total:
00:25.56 | eps:  0.596
  81 | Reward:   -51 | MA:  -125 | Steps:  133 Time:  0.48 | MA:  0.32 | Total:
00:26.04 | eps:  0.589
  82 | Reward:   -55 | MA:  -124 | Steps:  106 Time:  0.34 | MA:  0.32 | Total:
00:26.38 | eps:  0.584
  83 | Reward:   -20 | MA:  -123 | Steps:  103 Time:  0.34 | MA:  0.32 | Total:
00:26.72 | eps:  0.579
  84 | Reward:  -113 | MA:  -123 | Steps:  156 Time:  0.55 | MA:  0.32 | Total:
00:27.28 | eps:  0.572
  85 | Reward:   -36 | MA:  -122 | Steps:  122 Time:  0.40 | MA:  0.32 | Total:
00:27.68 | eps:  0.566
  86 | Reward:    -4 | MA:  -120 | Steps:  155 Time:  0.51 | MA:  0.32 | Total:
00:28.19 | eps:  0.559
  87 | Reward:   -40 | MA:  -119 | Steps:  111 Time:  0.41 | MA:  0.33 | Total:
00:28.60 | eps:  0.553
  88 | Reward:   -40 | MA:  -119 | Steps:  162 Time:  0.54 | MA:  0.33 | Total:
00:29.14 | eps:  0.546
  89 | Reward:   -10 | MA:  -117 | Steps:  176 Time:  0.58 | MA:  0.33 | Total:
00:29.72 | eps:  0.537
  90 | Reward:    -5 | MA:  -116 | Steps:  188 Time:  0.67 | MA:  0.33 | Total:
00:30.39 | eps:  0.528
  91 | Reward:   -70 | MA:  -116 | Steps:  141 Time:  0.46 | MA:  0.34 | Total:
00:30.85 | eps:  0.521
  92 | Reward:   -30 | MA:  -115 | Steps:  295 Time:  1.07 | MA:  0.34 | Total:
00:31.92 | eps:  0.507
  93 | Reward:  -175 | MA:  -115 | Steps:  199 Time:  0.67 | MA:  0.35 | Total:
00:32.59 | eps:  0.498
  94 | Reward:  -127 | MA:  -115 | Steps:  181 Time:  0.66 | MA:  0.35 | Total:
00:33.25 | eps:  0.489
  95 | Reward:    13 | MA:  -114 | Steps:  146 Time:  0.49 | MA:  0.35 | Total:
00:33.75 | eps:  0.482
```

```
  96 | Reward:     3 | MA:  -113 | Steps:  142 Time:  0.52 | MA:  0.35 | Total:
00:34.27 | eps:  0.475
  97 | Reward:   -60 | MA:  -112 | Steps:  144 Time:  0.49 | MA:  0.35 | Total:
00:34.76 | eps:  0.469
  98 | Reward:   -20 | MA:  -111 | Steps:  128 Time:  0.43 | MA:  0.36 | Total:
00:35.19 | eps:  0.462
  99 | Reward:   -71 | MA:  -111 | Steps:  309 Time:  1.12 | MA:  0.36 | Total:
00:36.31 | eps:  0.448
 100 | Reward:    31 | MA:  -109 | Steps:  188 Time:  1.94 | MA:  0.37 | Total:
00:38.25 | eps:  0.439
 101 | Reward:  -189 | MA:  -109 | Steps:  195 Time:  0.69 | MA:  0.38 | Total:
00:38.94 | eps:  0.429
 102 | Reward:   -15 | MA:  -107 | Steps:   96 Time:  0.33 | MA:  0.38 | Total:
00:39.27 | eps:  0.425
 103 | Reward:   -12 | MA:  -106 | Steps:  116 Time:  0.45 | MA:  0.38 | Total:
00:39.73 | eps:  0.419
 104 | Reward:  -183 | MA:  -106 | Steps:  253 Time:  0.88 | MA:  0.39 | Total:
00:40.60 | eps:  0.407
 105 | Reward:    19 | MA:  -104 | Steps:  999 Time:  4.12 | MA:  0.43 | Total:
00:44.72 | eps:  0.360
 106 | Reward:   -81 | MA:  -102 | Steps:  178 Time:  0.63 | MA:  0.44 | Total:
00:45.35 | eps:  0.351
 107 | Reward:  -116 | MA:  -102 | Steps:  453 Time:  1.81 | MA:  0.46 | Total:
00:47.16 | eps:  0.330
 108 | Reward:   129 | MA:   -99 | Steps:  999 Time:  4.19 | MA:  0.50 | Total:
00:51.35 | eps:  0.282
 109 | Reward:  -105 | MA:   -96 | Steps:  282 Time:  1.03 | MA:  0.51 | Total:
00:52.37 | eps:  0.269
 110 | Reward:  -181 | MA:   -97 | Steps:  612 Time:  2.42 | MA:  0.53 | Total:
00:54.80 | eps:  0.240
 111 | Reward:   177 | MA:   -95 | Steps:  320 Time:  1.23 | MA:  0.54 | Total:
00:56.03 | eps:  0.224
 112 | Reward:   -79 | MA:   -93 | Steps:  999 Time:  4.16 | MA:  0.58 | Total:
01:00.19 | eps:  0.177
 113 | Reward:   -99 | MA:   -90 | Steps:  999 Time:  4.40 | MA:  0.62 | Total:
01:04.58 | eps:  0.129
 114 | Reward:   -43 | MA:   -89 | Steps:  999 Time:  4.33 | MA:  0.66 | Total:
01:08.91 | eps:  0.082
 115 | Reward:   -35 | MA:   -87 | Steps:  999 Time:  4.63 | MA:  0.70 | Total:
01:13.54 | eps:  0.050
 116 | Reward:   -23 | MA:   -85 | Steps:  999 Time:  4.64 | MA:  0.75 | Total:
01:18.18 | eps:  0.049
 117 | Reward:   -41 | MA:   -83 | Steps:  999 Time:  4.72 | MA:  0.79 | Total:
01:22.90 | eps:  0.049
 118 | Reward:   -75 | MA:   -82 | Steps:  999 Time:  4.66 | MA:  0.83 | Total:
01:27.56 | eps:  0.048
 119 | Reward:   -25 | MA:   -79 | Steps:  999 Time:  4.82 | MA:  0.88 | Total:
01:32.38 | eps:  0.048
```

```
 120 | Reward:   -44 | MA:   -79 | Steps:  999 Time:  5.12 | MA:  0.93 | Total:
01:37.51 | eps:  0.047
 121 | Reward:   -37 | MA:   -78 | Steps:  999 Time:  5.00 | MA:  0.97 | Total:
01:42.51 | eps:  0.047
 122 | Reward:    -0 | MA:   -77 | Steps:  278 Time:  1.15 | MA:  0.98 | Total:
01:43.66 | eps:  0.046
 123 | Reward:    89 | MA:   -76 | Steps:  999 Time:  5.00 | MA:  1.03 | Total:
01:48.66 | eps:  0.046
 124 | Reward:   -17 | MA:   -75 | Steps:  269 Time:  1.06 | MA:  1.04 | Total:
01:49.72 | eps:  0.045
 125 | Reward:   161 | MA:   -70 | Steps:  999 Time:  4.44 | MA:  1.08 | Total:
01:54.16 | eps:  0.045
 126 | Reward:    -9 | MA:   -70 | Steps:  341 Time:  1.43 | MA:  1.09 | Total:
01:55.58 | eps:  0.044
 127 | Reward:   138 | MA:   -66 | Steps:  999 Time:  4.63 | MA:  1.14 | Total:
02:00.22 | eps:  0.044
 128 | Reward:   113 | MA:   -63 | Steps:  999 Time:  4.63 | MA:  1.18 | Total:
02:04.85 | eps:  0.043
 129 | Reward:   224 | MA:   -59 | Steps:  385 Time:  1.67 | MA:  1.19 | Total:
02:06.52 | eps:  0.043
 130 | Reward:   -20 | MA:   -58 | Steps:  462 Time:  2.00 | MA:  1.21 | Total:
02:08.52 | eps:  0.043
 131 | Reward:   205 | MA:   -53 | Steps:  389 Time:  1.64 | MA:  1.22 | Total:
02:10.16 | eps:  0.042
 132 | Reward:   223 | MA:   -50 | Steps:  640 Time:  2.91 | MA:  1.25 | Total:
02:13.06 | eps:  0.042
 133 | Reward:   -73 | MA:   -49 | Steps:  999 Time:  4.68 | MA:  1.29 | Total:
02:17.75 | eps:  0.041
 134 | Reward:   -21 | MA:   -49 | Steps:  534 Time:  2.40 | MA:  1.32 | Total:
02:20.14 | eps:  0.041
 135 | Reward:   -57 | MA:   -49 | Steps:  999 Time:  4.86 | MA:  1.36 | Total:
02:25.00 | eps:  0.041
 136 | Reward:   225 | MA:   -45 | Steps:  581 Time:  3.02 | MA:  1.39 | Total:
02:28.02 | eps:  0.040
 137 | Reward:   -61 | MA:   -44 | Steps:  999 Time:  4.84 | MA:  1.43 | Total:
02:32.87 | eps:  0.040
 138 | Reward:   -79 | MA:   -43 | Steps:  999 Time:  5.56 | MA:  1.49 | Total:
02:38.42 | eps:  0.039
 139 | Reward:   -20 | MA:   -42 | Steps:  999 Time:  5.32 | MA:  1.54 | Total:
02:43.75 | eps:  0.039
 140 | Reward:   -26 | MA:   -41 | Steps:  999 Time:  4.98 | MA:  1.58 | Total:
02:48.73 | eps:  0.039
 141 | Reward:     1 | MA:   -40 | Steps:  999 Time:  5.53 | MA:  1.64 | Total:
02:54.26 | eps:  0.038
 142 | Reward:   223 | MA:   -36 | Steps:  471 Time:  2.24 | MA:  1.66 | Total:
02:56.50 | eps:  0.038
 143 | Reward:   141 | MA:   -34 | Steps:  994 Time:  4.73 | MA:  1.70 | Total:
03:01.23 | eps:  0.037
```

```
 144 | Reward:   125 | MA:   -31 | Steps:  999 Time:  4.96 | MA:  1.75 | Total:
03:06.19 | eps:  0.037
 145 | Reward:    26 | MA:   -30 | Steps:  999 Time:  5.35 | MA:  1.80 | Total:
03:11.54 | eps:  0.037
 146 | Reward:    53 | MA:   -27 | Steps:  999 Time:  5.34 | MA:  1.85 | Total:
03:16.88 | eps:  0.036
 147 | Reward:   158 | MA:   -25 | Steps:  889 Time:  4.50 | MA:  1.89 | Total:
03:21.38 | eps:  0.036
 148 | Reward:   122 | MA:   -23 | Steps:  999 Time:  5.68 | MA:  1.94 | Total:
03:27.06 | eps:  0.036
 149 | Reward:  -109 | MA:   -24 | Steps:  342 Time:  1.67 | MA:  1.95 | Total:
03:28.73 | eps:  0.035
 150 | Reward:   208 | MA:   -21 | Steps:  852 Time:  9.43 | MA:  2.04 | Total:
03:38.16 | eps:  0.035
 151 | Reward:   139 | MA:   -20 | Steps:  777 Time:  4.48 | MA:  2.08 | Total:
03:42.64 | eps:  0.035
 152 | Reward:    60 | MA:   -18 | Steps:  999 Time:  5.82 | MA:  2.14 | Total:
03:48.46 | eps:  0.034
 153 | Reward:    61 | MA:   -17 | Steps:  999 Time:  6.37 | MA:  2.20 | Total:
03:54.83 | eps:  0.034
 154 | Reward:   -44 | MA:   -17 | Steps:  702 Time:  3.80 | MA:  2.23 | Total:
03:58.63 | eps:  0.033
 155 | Reward:   178 | MA:   -13 | Steps:  641 Time:  3.74 | MA:  2.26 | Total:
04:02.37 | eps:  0.033
 156 | Reward:   -63 | MA:   -13 | Steps:  421 Time:  2.27 | MA:  2.28 | Total:
04:04.64 | eps:  0.033
 157 | Reward:   228 | MA:   -10 | Steps:  352 Time:  1.75 | MA:  2.30 | Total:
04:06.38 | eps:  0.032
 158 | Reward:   -67 | MA:   -10 | Steps:  432 Time:  2.17 | MA:  2.32 | Total:
04:08.56 | eps:  0.032
 159 | Reward:   -68 | MA:   -10 | Steps:  739 Time:  3.70 | MA:  2.35 | Total:
04:12.25 | eps:  0.032
 160 | Reward:   186 | MA:    -8 | Steps:  600 Time:  3.30 | MA:  2.38 | Total:
04:15.56 | eps:  0.032
 161 | Reward:    76 | MA:    -6 | Steps:  999 Time:  5.44 | MA:  2.43 | Total:
04:21.00 | eps:  0.031
 162 | Reward:    43 | MA:    -6 | Steps:  999 Time:  5.56 | MA:  2.48 | Total:
04:26.56 | eps:  0.031
 163 | Reward:   174 | MA:    -3 | Steps:  593 Time:  2.95 | MA:  2.51 | Total:
04:29.51 | eps:  0.031
 164 | Reward:    41 | MA:    -2 | Steps:  999 Time:  5.64 | MA:  2.56 | Total:
04:35.14 | eps:  0.030
 165 | Reward:   169 | MA:     2 | Steps:  648 Time:  3.37 | MA:  2.59 | Total:
04:38.51 | eps:  0.030
 166 | Reward:     9 | MA:     3 | Steps:  999 Time:  5.37 | MA:  2.64 | Total:
04:43.88 | eps:  0.030
 167 | Reward:    94 | MA:     4 | Steps:  999 Time:  6.05 | MA:  2.70 | Total:
04:49.93 | eps:  0.029
```

```
 168 | Reward:   -20 | MA:     6 | Steps:  999 Time:  5.13 | MA:  2.75 | Total:
04:55.05 | eps:  0.029
 169 | Reward:   -68 | MA:     5 | Steps:  999 Time:  5.15 | MA:  2.79 | Total:
05:00.20 | eps:  0.029
 170 | Reward:   -25 | MA:     6 | Steps:  999 Time:  5.51 | MA:  2.85 | Total:
05:05.71 | eps:  0.029
 171 | Reward:    24 | MA:     6 | Steps:  999 Time:  5.30 | MA:  2.89 | Total:
05:11.01 | eps:  0.028
 172 | Reward:   275 | MA:     9 | Steps:  327 Time:  1.73 | MA:  2.91 | Total:
05:12.74 | eps:  0.028
 173 | Reward:   228 | MA:    12 | Steps:  761 Time:  4.40 | MA:  2.95 | Total:
05:17.14 | eps:  0.028
 174 | Reward:   149 | MA:    13 | Steps:  999 Time:  5.41 | MA:  3.00 | Total:
05:22.56 | eps:  0.027
 175 | Reward:    28 | MA:    14 | Steps:  999 Time:  5.56 | MA:  3.05 | Total:
05:28.11 | eps:  0.027
 176 | Reward:   110 | MA:    16 | Steps:  999 Time:  5.50 | MA:  3.10 | Total:
05:33.62 | eps:  0.027
 177 | Reward:   128 | MA:    17 | Steps:  999 Time:  5.79 | MA:  3.15 | Total:
05:39.41 | eps:  0.027
 178 | Reward:   141 | MA:    21 | Steps:  842 Time:  4.96 | MA:  3.20 | Total:
05:44.37 | eps:  0.026
 179 | Reward:   121 | MA:    22 | Steps:  999 Time:  5.98 | MA:  3.25 | Total:
05:50.36 | eps:  0.026
 180 | Reward:   152 | MA:    25 | Steps:  999 Time:  5.65 | MA:  3.30 | Total:
05:56.00 | eps:  0.026
 181 | Reward:   245 | MA:    28 | Steps:  613 Time:  3.42 | MA:  3.33 | Total:
05:59.42 | eps:  0.026
 182 | Reward:   142 | MA:    30 | Steps:  999 Time:  5.74 | MA:  3.39 | Total:
06:05.16 | eps:  0.025
 183 | Reward:   199 | MA:    32 | Steps:  450 Time:  2.48 | MA:  3.41 | Total:
06:07.64 | eps:  0.025
 184 | Reward:   106 | MA:    34 | Steps:  999 Time:  6.00 | MA:  3.46 | Total:
06:13.64 | eps:  0.025
 185 | Reward:   118 | MA:    36 | Steps:  999 Time:  5.92 | MA:  3.52 | Total:
06:19.55 | eps:  0.025
 186 | Reward:    75 | MA:    36 | Steps:  999 Time:  5.86 | MA:  3.57 | Total:
06:25.41 | eps:  0.024
 187 | Reward:    91 | MA:    38 | Steps:  999 Time:  6.49 | MA:  3.63 | Total:
06:31.90 | eps:  0.024
 188 | Reward:   239 | MA:    40 | Steps:  552 Time:  3.31 | MA:  3.66 | Total:
06:35.21 | eps:  0.024
 189 | Reward:   151 | MA:    42 | Steps:  999 Time:  6.20 | MA:  3.72 | Total:
06:41.41 | eps:  0.024
 190 | Reward:   128 | MA:    43 | Steps:  999 Time:  6.52 | MA:  3.78 | Total:
06:47.92 | eps:  0.023
 191 | Reward:   142 | MA:    45 | Steps:  999 Time:  6.88 | MA:  3.84 | Total:
06:54.81 | eps:  0.023
```

```
 192 | Reward:   135 | MA:    47 | Steps:  999 Time:  6.44 | MA:  3.89 | Total:
07:01.25 | eps:  0.023
 193 | Reward:   157 | MA:    50 | Steps:  999 Time:  6.55 | MA:  3.95 | Total:
07:07.80 | eps:  0.023
 194 | Reward:   100 | MA:    53 | Steps:  999 Time:  6.26 | MA:  4.01 | Total:
07:14.07 | eps:  0.022
 195 | Reward:   125 | MA:    54 | Steps:  999 Time:  6.77 | MA:  4.07 | Total:
07:20.84 | eps:  0.022
 196 | Reward:   109 | MA:    55 | Steps:  999 Time:  6.97 | MA:  4.14 | Total:
07:27.81 | eps:  0.022
 197 | Reward:   122 | MA:    57 | Steps:  999 Time:  6.83 | MA:  4.20 | Total:
07:34.64 | eps:  0.022
 198 | Reward:   219 | MA:    59 | Steps:  825 Time:  5.48 | MA:  4.25 | Total:
07:40.12 | eps:  0.022
 199 | Reward:   165 | MA:    61 | Steps:  999 Time:  6.45 | MA:  4.30 | Total:
07:46.57 | eps:  0.021
 200 | Reward:   118 | MA:    62 | Steps:  999 Time: 12.55 | MA:  4.41 | Total:
07:59.13 | eps:  0.021
 201 | Reward:   148 | MA:    66 | Steps:  999 Time:  6.87 | MA:  4.47 | Total:
08:05.99 | eps:  0.021
 202 | Reward:   127 | MA:    67 | Steps:  999 Time:  6.80 | MA:  4.54 | Total:
08:12.80 | eps:  0.021
 203 | Reward:   126 | MA:    68 | Steps:  999 Time:  6.96 | MA:  4.60 | Total:
08:19.76 | eps:  0.020
 204 | Reward:   162 | MA:    72 | Steps:  999 Time:  6.60 | MA:  4.66 | Total:
08:26.36 | eps:  0.020
 205 | Reward:    83 | MA:    73 | Steps:  999 Time:  6.62 | MA:  4.68 | Total:
08:32.98 | eps:  0.020
 206 | Reward:   231 | MA:    76 | Steps:  706 Time:  5.00 | MA:  4.73 | Total:
08:37.98 | eps:  0.020
 207 | Reward:   252 | MA:    79 | Steps:  355 Time:  2.34 | MA:  4.73 | Total:
08:40.33 | eps:  0.020
 208 | Reward:    72 | MA:    79 | Steps:  999 Time:  6.71 | MA:  4.76 | Total:
08:47.04 | eps:  0.019
 209 | Reward:   182 | MA:    82 | Steps:  999 Time:  7.10 | MA:  4.82 | Total:
08:54.14 | eps:  0.019
 210 | Reward:   -23 | MA:    83 | Steps:  438 Time:  3.00 | MA:  4.82 | Total:
08:57.14 | eps:  0.019
 211 | Reward:   133 | MA:    83 | Steps:  999 Time:  7.23 | MA:  4.88 | Total:
09:04.36 | eps:  0.019
 212 | Reward:   140 | MA:    85 | Steps:  999 Time:  7.24 | MA:  4.91 | Total:
09:11.61 | eps:  0.019
 213 | Reward:   139 | MA:    87 | Steps:  999 Time:  6.78 | MA:  4.94 | Total:
09:18.39 | eps:  0.019
 214 | Reward:   125 | MA:    89 | Steps:  999 Time:  7.19 | MA:  4.97 | Total:
09:25.57 | eps:  0.018
 215 | Reward:   174 | MA:    91 | Steps:  999 Time:  7.59 | MA:  5.00 | Total:
09:33.16 | eps:  0.018
```

```
 216 | Reward:   132 | MA:    93 | Steps:  999 Time:  7.09 | MA:  5.02 | Total:
09:40.25 | eps:  0.018
 217 | Reward:   125 | MA:    94 | Steps:  999 Time:  7.25 | MA:  5.05 | Total:
09:47.50 | eps:  0.018
 218 | Reward:   134 | MA:    96 | Steps:  999 Time:  7.59 | MA:  5.08 | Total:
09:55.09 | eps:  0.018
 219 | Reward:   102 | MA:    98 | Steps:  999 Time:  6.82 | MA:  5.10 | Total:
10:01.91 | eps:  0.017
 220 | Reward:   128 | MA:    99 | Steps:  999 Time:  6.95 | MA:  5.11 | Total:
10:08.86 | eps:  0.017
 221 | Reward:   118 | MA:   101 | Steps:  999 Time:  6.92 | MA:  5.13 | Total:
10:15.78 | eps:  0.017
 222 | Reward:   130 | MA:   102 | Steps:  999 Time:  7.08 | MA:  5.19 | Total:
10:22.86 | eps:  0.017
 223 | Reward:   118 | MA:   103 | Steps:  999 Time:  7.07 | MA:  5.21 | Total:
10:29.94 | eps:  0.017
 224 | Reward:   156 | MA:   104 | Steps:  999 Time:  6.99 | MA:  5.27 | Total:
10:36.93 | eps:  0.017
 225 | Reward:   232 | MA:   105 | Steps:  857 Time:  6.28 | MA:  5.29 | Total:
10:43.21 | eps:  0.016
 226 | Reward:   147 | MA:   107 | Steps:  999 Time:  7.17 | MA:  5.35 | Total:
10:50.38 | eps:  0.016
 227 | Reward:   152 | MA:   107 | Steps:  999 Time:  7.50 | MA:  5.38 | Total:
10:57.87 | eps:  0.016
 228 | Reward:   113 | MA:   107 | Steps:  999 Time:  7.07 | MA:  5.40 | Total:
11:04.94 | eps:  0.016
 229 | Reward:   233 | MA:   107 | Steps:  952 Time:  6.45 | MA:  5.45 | Total:
11:11.40 | eps:  0.016
 230 | Reward:   233 | MA:   109 | Steps:  323 Time:  2.12 | MA:  5.45 | Total:
11:13.51 | eps:  0.016
 231 | Reward:   172 | MA:   109 | Steps:  999 Time:  6.97 | MA:  5.50 | Total:
11:20.48 | eps:  0.015
 232 | Reward:   110 | MA:   108 | Steps:  999 Time:  7.27 | MA:  5.55 | Total:
11:27.75 | eps:  0.015
 233 | Reward:   173 | MA:   110 | Steps:  999 Time:  7.01 | MA:  5.57 | Total:
11:34.76 | eps:  0.015
 234 | Reward:   143 | MA:   112 | Steps:  999 Time:  7.21 | MA:  5.62 | Total:
11:41.97 | eps:  0.015
 235 | Reward:   239 | MA:   115 | Steps:  419 Time:  2.81 | MA:  5.60 | Total:
11:44.78 | eps:  0.015
 236 | Reward:   225 | MA:   115 | Steps:  517 Time:  3.53 | MA:  5.60 | Total:
11:48.32 | eps:  0.015
 237 | Reward:   224 | MA:   118 | Steps:  530 Time:  3.96 | MA:  5.59 | Total:
11:52.28 | eps:  0.015
 238 | Reward:   158 | MA:   120 | Steps:  999 Time:  7.45 | MA:  5.61 | Total:
11:59.73 | eps:  0.014
 239 | Reward:   296 | MA:   123 | Steps:  435 Time:  3.04 | MA:  5.59 | Total:
12:02.77 | eps:  0.014
```

```
 240 | Reward:   222 | MA:   126 | Steps:  344 Time:  2.33 | MA:  5.56 | Total:
12:05.10 | eps:  0.014
 241 | Reward:   268 | MA:   128 | Steps:  290 Time:  1.98 | MA:  5.53 | Total:
12:07.09 | eps:  0.014
 242 | Reward:   247 | MA:   129 | Steps:  455 Time:  3.13 | MA:  5.54 | Total:
12:10.22 | eps:  0.014
 243 | Reward:   241 | MA:   130 | Steps:  368 Time:  2.53 | MA:  5.52 | Total:
12:12.74 | eps:  0.014
 244 | Reward:   268 | MA:   131 | Steps:  351 Time:  2.40 | MA:  5.49 | Total:
12:15.14 | eps:  0.014
 245 | Reward:   285 | MA:   134 | Steps:  542 Time:  3.88 | MA:  5.47 | Total:
12:19.02 | eps:  0.013
 246 | Reward:   -64 | MA:   133 | Steps:  379 Time:  2.62 | MA:  5.45 | Total:
12:21.64 | eps:  0.013
 247 | Reward:   228 | MA:   133 | Steps:  464 Time:  3.32 | MA:  5.44 | Total:
12:24.96 | eps:  0.013
 248 | Reward:   237 | MA:   134 | Steps:  346 Time:  2.38 | MA:  5.40 | Total:
12:27.34 | eps:  0.013
 249 | Reward:   275 | MA:   138 | Steps:  493 Time:  3.44 | MA:  5.42 | Total:
12:30.78 | eps:  0.013
 250 | Reward:   206 | MA:   138 | Steps:  311 Time:  4.14 | MA:  5.37 | Total:
12:34.92 | eps:  0.013
 251 | Reward:   167 | MA:   138 | Steps:  999 Time:  7.45 | MA:  5.40 | Total:
12:42.37 | eps:  0.013
 252 | Reward:   268 | MA:   141 | Steps:  259 Time:  1.84 | MA:  5.36 | Total:
12:44.20 | eps:  0.013
 253 | Reward:   242 | MA:   142 | Steps:  553 Time:  4.13 | MA:  5.34 | Total:
12:48.34 | eps:  0.012
 254 | Reward:   286 | MA:   146 | Steps:  243 Time:  1.75 | MA:  5.31 | Total:
12:50.08 | eps:  0.012
 255 | Reward:   256 | MA:   146 | Steps:  499 Time:  3.67 | MA:  5.31 | Total:
12:53.75 | eps:  0.012
 256 | Reward:   260 | MA:   150 | Steps:  327 Time:  2.57 | MA:  5.32 | Total:
12:56.32 | eps:  0.012
 257 | Reward:   276 | MA:   150 | Steps:  371 Time:  2.96 | MA:  5.33 | Total:
12:59.28 | eps:  0.012
 258 | Reward:   257 | MA:   153 | Steps:  368 Time:  2.84 | MA:  5.34 | Total:
13:02.13 | eps:  0.012
 259 | Reward:   244 | MA:   157 | Steps:  329 Time:  2.43 | MA:  5.32 | Total:
13:04.56 | eps:  0.012
 260 | Reward:   281 | MA:   157 | Steps:  291 Time:  2.06 | MA:  5.31 | Total:
13:06.62 | eps:  0.012
 261 | Reward:   254 | MA:   159 | Steps:  342 Time:  2.51 | MA:  5.28 | Total:
13:09.13 | eps:  0.011
 262 | Reward:   223 | MA:   161 | Steps:  306 Time:  2.20 | MA:  5.25 | Total:
13:11.34 | eps:  0.011
 263 | Reward:   270 | MA:   162 | Steps:  298 Time:  2.17 | MA:  5.24 | Total:
13:13.50 | eps:  0.011
```

```
 264 | Reward:   235 | MA:   164 | Steps:  302 Time:  2.18 | MA:  5.21 | Total:
13:15.68 | eps:  0.011
 265 | Reward:   276 | MA:   165 | Steps:  244 Time:  1.79 | MA:  5.19 | Total:
13:17.47 | eps:  0.011
 266 | Reward:   246 | MA:   167 | Steps:  495 Time:  3.64 | MA:  5.17 | Total:
13:21.11 | eps:  0.011
 267 | Reward:   251 | MA:   169 | Steps:  299 Time:  2.21 | MA:  5.13 | Total:
13:23.32 | eps:  0.011
 268 | Reward:   278 | MA:   172 | Steps:  347 Time:  2.56 | MA:  5.11 | Total:
13:25.87 | eps:  0.011
 269 | Reward:   294 | MA:   176 | Steps:  272 Time:  1.97 | MA:  5.08 | Total:
13:27.85 | eps:  0.011
 270 | Reward:   246 | MA:   178 | Steps:  374 Time:  2.86 | MA:  5.05 | Total:
13:30.71 | eps:  0.010
 271 | Reward:   227 | MA:   180 | Steps:  298 Time:  2.23 | MA:  5.02 | Total:
13:32.94 | eps:  0.010
 272 | Reward:   254 | MA:   180 | Steps:  411 Time:  3.11 | MA:  5.03 | Total:
13:36.05 | eps:  0.010
 273 | Reward:   229 | MA:   180 | Steps:  297 Time:  2.25 | MA:  5.01 | Total:
13:38.30 | eps:  0.010
 274 | Reward:   225 | MA:   181 | Steps:  463 Time:  3.53 | MA:  4.99 | Total:
13:41.83 | eps:  0.010
 275 | Reward:   282 | MA:   183 | Steps:  350 Time:  2.57 | MA:  4.96 | Total:
13:44.39 | eps:  0.010
 276 | Reward:   254 | MA:   185 | Steps:  379 Time:  2.89 | MA:  4.94 | Total:
13:47.28 | eps:  0.010
 277 | Reward:   235 | MA:   186 | Steps:  361 Time:  2.77 | MA:  4.91 | Total:
13:50.05 | eps:  0.010
 278 | Reward:   244 | MA:   187 | Steps:  387 Time:  2.97 | MA:  4.89 | Total:
13:53.02 | eps:  0.010
 279 | Reward:   284 | MA:   189 | Steps:  321 Time:  2.39 | MA:  4.85 | Total:
13:55.41 | eps:  0.010
 280 | Reward:   207 | MA:   189 | Steps:  583 Time:  4.52 | MA:  4.84 | Total:
13:59.93 | eps:  0.009
 281 | Reward:   248 | MA:   189 | Steps:  442 Time:  3.51 | MA:  4.84 | Total:
14:03.44 | eps:  0.009
 282 | Reward:   262 | MA:   190 | Steps:  311 Time:  2.48 | MA:  4.81 | Total:
14:05.92 | eps:  0.009
 283 | Reward:   266 | MA:   191 | Steps:  408 Time:  3.13 | MA:  4.81 | Total:
14:09.05 | eps:  0.009
 284 | Reward:   247 | MA:   192 | Steps:  377 Time:  2.99 | MA:  4.78 | Total:
14:12.03 | eps:  0.009
 285 | Reward:   230 | MA:   194 | Steps:  315 Time:  2.38 | MA:  4.75 | Total:
14:14.42 | eps:  0.009
 286 | Reward:   269 | MA:   195 | Steps:  370 Time:  3.28 | MA:  4.72 | Total:
14:17.70 | eps:  0.009
 287 | Reward:   302 | MA:   198 | Steps:  350 Time:  2.64 | MA:  4.68 | Total:
14:20.33 | eps:  0.009
```

```
 288 | Reward:   275 | MA:   198 | Steps:  424 Time:  3.28 | MA:  4.68 | Total:
14:23.61 | eps:  0.009
 289 | Reward:   231 | MA:   199 | Steps:  324 Time:  2.49 | MA:  4.65 | Total:
14:26.10 | eps:  0.009
 290 | Reward:   278 | MA:   200 | Steps:  262 Time:  2.01 | MA:  4.60 | Total:
14:28.11 | eps:  0.009
```

1.6.7 Store Results

[17]:
```
results = pd.DataFrame({'episode': list(range(1, len(episode_steps) + 1)),
                        'steps'  : episode_steps,
                        'rewards': episode_rewards,
                        'time'   : episode_time,
                        'epsilon': episode_eps})

fn = 'results.csv' if train else 'performance.csv'
results.to_csv(experiment_dir / fn, index=False)
```

[ ]:
```
params = dict(gamma=gamma, layers=layers, learning_rate=learning_rate,␣
↪l2_reg=l2_reg,
              tau=tau, replay_capacity=replay_capacity,␣
↪minibatch_size=minibatch_size,
              epsilon_start=epsilon_start, epsilon_end=epsilon_end,␣
↪epsilon_linear_steps=epsilon_linear_steps,
              epsilon_exp_decay=epsilon_exp_decay)

parameters = pd.Series(params).to_frame().reset_index()
parameters.columns = ['parameter', 'value']
parameters.to_csv(experiment_dir / 'parameters.csv', index=False)
```

1.6.8 Plot Results

[7]:
```
results = pd.read_csv(experiment_dir / 'results.csv').rename(columns=str.
↪capitalize)
results = results.set_index('Episode')
results['MA100'] = results.rolling(window=100, min_periods=25).Rewards.mean()
```

[8]:
```
results.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 291 entries, 1 to 291
Data columns (total 5 columns):
Epsilon    291 non-null float64
Rewards    291 non-null float64
Steps      291 non-null int64
Time       291 non-null float64
MA100      267 non-null float64
```

```
dtypes: float64(4), int64(1)
memory usage: 13.6 KB
```

[9]:
```
fig, axes = plt.subplots(ncols=2, figsize=(14, 4))
results[['Rewards', 'MA100']].plot(ax=axes[0])
axes[0].set_ylabel('Rewards')
results[['Steps', 'Epsilon']].plot(secondary_y='Epsilon', ax=axes[1]);
fig.suptitle('Double Deep Q-Network Agent | Lunar Lander', fontsize=16)
fig.tight_layout()
fig.subplots_adjust(top=.9)
fig.savefig('figures/ddqn_lunarlander', dpi=300)
```