reinforcement_learning_assignment (1)

July 10, 2023

Install the required libraries

```
[ ]: # !pip install tensorflow==2.3.0
     # !pip install gym
     !pip install keras
     !pip install keras-rl2
     !pip install 'gym[classic_control]'
```

Import the necessary libraries and set up the CartPole environment

```
[ ]: import gym
     import random
     import numpy as np
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense, Flatten, Activation
     from tensorflow.keras.optimizers import Adam
     import os
     from rl.agents import DQNAgent
     from rl.policy import BoltzmannQPolicy
     from rl.memory import SequentialMemory
```

```
[ ]: env = gym.make('CartPole-v1')
     states = env.observation_space.shape[0]
     actions = env.action_space.n
```

```
/usr/local/lib/python3.8/dist-packages/gym/core.py:317: DeprecationWarning:
WARN: Initializing wrapper in old step API which returns one bool instead

of two. It is recommended to set `new_step_api=True` to use new step API. This

will be the default behaviour in future.
  deprecation(
/usr/local/lib/python3.8/dist-
packages/gym/wrappers/step_api_compatibility.py:39: DeprecationWarning:
WARN: Initializing environment in old step API which returns one bool

instead of two. It is recommended to set `new_step_api=True` to use new step

API. This will be the default behaviour in future.
  deprecation(
```

[ ]:
```
actions
```

[ ]:
```
2
```

[ ]:
```
os.environ['SDL_VIDEODRIVER']='dummy'
```

[ ]:
```
episodes = 20
for episode in range(1, episodes+1):
    state = env.reset()
    done = False
    score = 0

    while not done:
        env.render()
        action = random.choice([0,1])
        n_state, reward, done, info = env.step(action)
        score+=reward
    print('Episode:{} Score:{}'.format(episode, score))
```

```
/usr/local/lib/python3.8/dist-packages/gym/core.py:49: DeprecationWarning:
WARN: You are calling render method, but you didn't specified the argument

render_mode at environment initialization. To maintain backward compatibility,

the environment will render in human mode.

If you want to render in human mode, initialize the environment in this way:

gym.make('EnvName', render_mode='human') and don't call the render method.

See here for more information: https://www.gymlibrary.ml/content/api/
  deprecation(
```

```
Episode:1 Score:14.0
Episode:2 Score:23.0
Episode:3 Score:21.0
Episode:4 Score:21.0
Episode:5 Score:16.0
Episode:6 Score:105.0
Episode:7 Score:12.0
Episode:8 Score:58.0
Episode:9 Score:15.0
Episode:10 Score:15.0
Episode:11 Score:14.0
Episode:12 Score:27.0
Episode:13 Score:19.0
Episode:14 Score:13.0
Episode:15 Score:19.0
Episode:16 Score:31.0
Episode:17 Score:25.0
```

```
Episode:18 Score:16.0
Episode:19 Score:29.0
Episode:20 Score:17.0
```

Define the model

Next, define a function that will create the model. This model will take in the current state of the environment as input and output the action to be taken.

[ ]:
```
def build_model(states, actions):
    model = Sequential()
    model.add(Flatten(input_shape=(1,states)))
    model.add(Dense(32, activation='relu'))
    model.add(Dense(32, activation='relu'))
    model.add(Dense(actions, activation='linear'))
    return model
```

[ ]:
```
model = build_model(states, actions)
```

[ ]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 4)                 0

 dense (Dense)               (None, 32)                160

 dense_1 (Dense)             (None, 32)                1056

 dense_2 (Dense)             (None, 2)                 66

=================================================================
Total params: 1,282
Trainable params: 1,282
Non-trainable params: 0
_________________________________________________________________
```

Build the agent and train the model

[ ]:
```
def build_agent(model, actions):
    policy = BoltzmannQPolicy()
    memory = SequentialMemory(limit=50000, window_length=1)
    dqn = DQNAgent(model=model, memory=memory, policy=policy,␣
 ↪nb_actions=actions, nb_steps_warmup=10, target_model_update=1e-2)
    return dqn
```

[ ]:
```python
dqn = build_agent(model, actions)
dqn.compile(Adam(lr=1e-3), metrics=['mae'])
dqn.fit(env, nb_steps=50000, visualize=True, verbose=1)
```

```
/usr/local/lib/python3.8/dist-
packages/keras/optimizers/optimizer_v2/adam.py:110: UserWarning: The `lr`
argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)

Training for 50000 steps …
Interval 1 (0 steps performed)
    4/10000 […] - ETA: 3:23 - reward: 1.0000

/usr/local/lib/python3.8/dist-packages/keras/engine/training_v1.py:2067:
UserWarning: `Model.state_updates` will be removed in a future version. This
property should not be used in TensorFlow 2.0, as `updates` are applied
automatically.
  updates=self.state_updates,
/usr/local/lib/python3.8/dist-packages/gym/core.py:43: DeprecationWarning:
WARN: The argument mode in render method is deprecated; use render_mode

during environment initialization instead.

See here for more information: https://www.gymlibrary.ml/content/api/
  deprecation(

   10/10000 […] - ETA: 3:21 - reward: 1.0000

/usr/local/lib/python3.8/dist-packages/rl/memory.py:37: UserWarning: Not enough
entries to sample without replacement. Consider increasing your warm-up phase to
avoid oversampling!
  warnings.warn('Not enough entries to sample without replacement. Consider
increasing your warm-up phase to avoid oversampling!')
/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 10 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)

   21/10000 […] - ETA: 8:59 - reward: 1.0000

/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 11 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)
/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 12 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)
/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 13 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)
/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 14 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)
```

```
/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 15 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)
/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 16 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)
/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 17 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)
/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 18 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)
/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 19 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)
/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 20 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)
/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 21 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)

   30/10000 […] - ETA: 7:16 - reward: 1.0000

/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 22 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)
/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 23 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)
/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 24 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)
/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 25 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)
/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 26 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)
/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 27 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)
/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 28 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)
/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 29 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)
/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
```

```
function is deprecated. Please call randint(1, 30 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)
/usr/local/lib/python3.8/dist-packages/rl/memory.py:38: DeprecationWarning: This
function is deprecated. Please call randint(1, 31 + 1) instead
  batch_idxs = np.random.random_integers(low, high - 1, size=size)

10000/10000 [==============================] - 203s 20ms/step - reward: 1.0000
82 episodes - episode_reward: 120.902 [8.000, 500.000] - loss: 2.475 - mae:
21.068 - mean_q: 42.770

Interval 2 (10000 steps performed)
10000/10000 [==============================] - 202s 20ms/step - reward: 1.0000
30 episodes - episode_reward: 328.533 [257.000, 500.000] - loss: 4.775 - mae:
46.245 - mean_q: 93.583

Interval 3 (20000 steps performed)
10000/10000 [==============================] - 202s 20ms/step - reward: 1.0000
28 episodes - episode_reward: 349.393 [279.000, 500.000] - loss: 4.910 - mae:
52.077 - mean_q: 105.062

Interval 4 (30000 steps performed)
10000/10000 [==============================] - 202s 20ms/step - reward: 1.0000
25 episodes - episode_reward: 416.800 [315.000, 500.000] - loss: 4.627 - mae:
53.337 - mean_q: 107.454

Interval 5 (40000 steps performed)
10000/10000 [==============================] - 202s 20ms/step - reward: 1.0000
done, took 1011.950 seconds
```

```
[ ]: <keras.callbacks.History at 0x7fc02face790>
```

```
[ ]: scores = dqn.test(env, nb_episodes=100, visualize=True)
     print(np.mean(scores.history['episode_reward']))
```

```
Testing for 100 episodes …
Episode 1: reward: 337.000, steps: 337
Episode 2: reward: 500.000, steps: 500
Episode 3: reward: 400.000, steps: 400
Episode 4: reward: 430.000, steps: 430
Episode 5: reward: 438.000, steps: 438
Episode 6: reward: 443.000, steps: 443
Episode 7: reward: 452.000, steps: 452
Episode 8: reward: 410.000, steps: 410
Episode 9: reward: 366.000, steps: 366
Episode 10: reward: 452.000, steps: 452
Episode 11: reward: 431.000, steps: 431
Episode 12: reward: 384.000, steps: 384
Episode 13: reward: 389.000, steps: 389
Episode 14: reward: 359.000, steps: 359
Episode 15: reward: 420.000, steps: 420
Episode 16: reward: 364.000, steps: 364
Episode 17: reward: 448.000, steps: 448
Episode 18: reward: 344.000, steps: 344
Episode 19: reward: 431.000, steps: 431
Episode 20: reward: 372.000, steps: 372
Episode 21: reward: 344.000, steps: 344
Episode 22: reward: 467.000, steps: 467
Episode 23: reward: 468.000, steps: 468
Episode 24: reward: 452.000, steps: 452
Episode 25: reward: 439.000, steps: 439
Episode 26: reward: 409.000, steps: 409
Episode 27: reward: 434.000, steps: 434
Episode 28: reward: 377.000, steps: 377
Episode 29: reward: 477.000, steps: 477
Episode 30: reward: 479.000, steps: 479
Episode 31: reward: 393.000, steps: 393
Episode 32: reward: 464.000, steps: 464
Episode 33: reward: 370.000, steps: 370
Episode 34: reward: 429.000, steps: 429
Episode 35: reward: 340.000, steps: 340
Episode 36: reward: 382.000, steps: 382
Episode 37: reward: 394.000, steps: 394
Episode 38: reward: 477.000, steps: 477
Episode 39: reward: 462.000, steps: 462
Episode 40: reward: 436.000, steps: 436
Episode 41: reward: 393.000, steps: 393
Episode 42: reward: 364.000, steps: 364
Episode 43: reward: 484.000, steps: 484
Episode 44: reward: 500.000, steps: 500
Episode 45: reward: 428.000, steps: 428
Episode 46: reward: 399.000, steps: 399
Episode 47: reward: 500.000, steps: 500
Episode 48: reward: 448.000, steps: 448
Episode 49: reward: 476.000, steps: 476
Episode 50: reward: 491.000, steps: 491
Episode 51: reward: 496.000, steps: 496
Episode 52: reward: 417.000, steps: 417
Episode 53: reward: 408.000, steps: 408
Episode 54: reward: 435.000, steps: 435
Episode 55: reward: 485.000, steps: 485
Episode 56: reward: 436.000, steps: 436
Episode 57: reward: 415.000, steps: 415
Episode 58: reward: 500.000, steps: 500
Episode 59: reward: 331.000, steps: 331
Episode 60: reward: 500.000, steps: 500
Episode 61: reward: 358.000, steps: 358
Episode 62: reward: 493.000, steps: 493
Episode 63: reward: 369.000, steps: 369
Episode 64: reward: 357.000, steps: 357
Episode 65: reward: 431.000, steps: 431
Episode 66: reward: 360.000, steps: 360
Episode 67: reward: 340.000, steps: 340
Episode 68: reward: 442.000, steps: 442
Episode 69: reward: 423.000, steps: 423
Episode 70: reward: 467.000, steps: 467
Episode 71: reward: 444.000, steps: 444
Episode 72: reward: 408.000, steps: 408
Episode 73: reward: 328.000, steps: 328
Episode 74: reward: 393.000, steps: 393
Episode 75: reward: 409.000, steps: 409
Episode 76: reward: 427.000, steps: 427
Episode 77: reward: 338.000, steps: 338
Episode 78: reward: 435.000, steps: 435
Episode 79: reward: 500.000, steps: 500
Episode 80: reward: 500.000, steps: 500
Episode 81: reward: 352.000, steps: 352
Episode 82: reward: 421.000, steps: 421
Episode 83: reward: 384.000, steps: 384
Episode 84: reward: 469.000, steps: 469
Episode 85: reward: 333.000, steps: 333
Episode 86: reward: 467.000, steps: 467
Episode 87: reward: 396.000, steps: 396
Episode 88: reward: 339.000, steps: 339
Episode 89: reward: 362.000, steps: 362
Episode 90: reward: 415.000, steps: 415
Episode 91: reward: 437.000, steps: 437
Episode 92: reward: 398.000, steps: 398
Episode 93: reward: 465.000, steps: 465
Episode 94: reward: 500.000, steps: 500
Episode 95: reward: 500.000, steps: 500
Episode 96: reward: 372.000, steps: 372
Episode 97: reward: 500.000, steps: 500
Episode 98: reward: 456.000, steps: 456
Episode 99: reward: 390.000, steps: 390
Episode 100: reward: 408.000, steps: 408
421.55
```

```
[ ]: _ = dqn.test(env, nb_episodes=15, visualize=True)
```

```
Testing for 15 episodes …
Episode 1: reward: 462.000, steps: 462
Episode 2: reward: 394.000, steps: 394
Episode 3: reward: 489.000, steps: 489
Episode 4: reward: 472.000, steps: 472
Episode 5: reward: 500.000, steps: 500
Episode 6: reward: 500.000, steps: 500
Episode 7: reward: 422.000, steps: 422
Episode 8: reward: 500.000, steps: 500
Episode 9: reward: 466.000, steps: 466
Episode 10: reward: 463.000, steps: 463
Episode 11: reward: 385.000, steps: 385
Episode 12: reward: 352.000, steps: 352
Episode 13: reward: 418.000, steps: 418
Episode 14: reward: 387.000, steps: 387
Episode 15: reward: 475.000, steps: 475
```

[ ]:
```
dqn.save_weights('dqn_weights.h5f', overwrite=True)
```

[ ]:
```
del model
del dqn
del env
```

Test the trained model

Now that I have trained the model, I can test it by running it on the CartPole environment and see how well it performs.

[ ]:
```
env = gym.make('CartPole-v1')
actions = env.action_space.n
states = env.observation_space.shape[0]
model = build_model(states, actions)
dqn = build_agent(model, actions)
dqn.compile(Adam(lr=1e-3), metrics=['mae'])
```

[ ]:
```
dqn.load_weights('dqn_weights.h5f')
```

[ ]:
```
_ = dqn.test(env, nb_episodes=10, visualize=True)
```

```
Testing for 10 episodes …
Episode 1: reward: 372.000, steps: 372
Episode 2: reward: 385.000, steps: 385
Episode 3: reward: 457.000, steps: 457
Episode 4: reward: 365.000, steps: 365
Episode 5: reward: 400.000, steps: 400
Episode 6: reward: 458.000, steps: 458
Episode 7: reward: 404.000, steps: 404
Episode 8: reward: 406.000, steps: 406
Episode 9: reward: 390.000, steps: 390
Episode 10: reward: 463.000, steps: 463
```

[ ]:
```

```