UDACITY

Return to Classroom

Collaboration and Competition

REVIEW

CODE REVIEW

HISTORY

Requires Changes

2 specifications require changes

Congratulations on your last two agents.

You made a great submission that solved this quite unstable environment in less than a 100 episodes. It's common for this environment to give variant results every run, but your model is stable. Give yourself proper credit.

You did a great job with the hard part. Only one requirement's left to do. Please add:

- a condition to stop training after at least 100 episodes have passed.

I hope this nanodegree gives you a good kick towards your dreams. These resources will give you a different point of view for Deep Reinforcement Learning:

- MIT 6.S094: Deep Reinforcement Learning video.
- Deep Reinforcement Learning by David Silver.

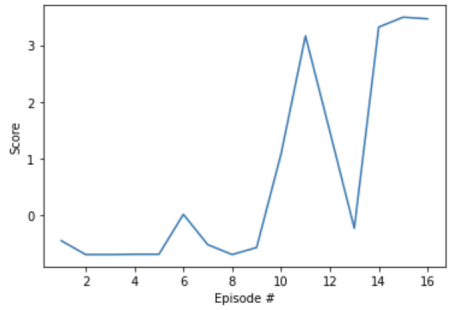
Training Code

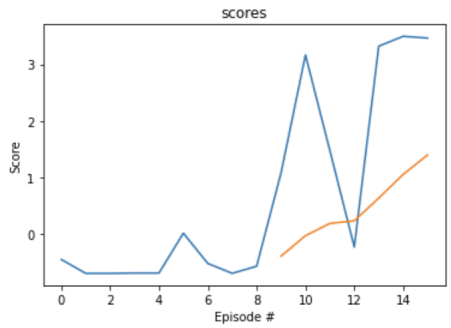
The repository includes functional, well-documented, and organized code for training the agent.

Your code functions correctly. Your model is so well built that your agents reach the required score before the first 100 episodes. Here are the results I got when I ran your code.

```
: 1 scores = ddpg(target_mean_scores_deque=0.5)
  2
  3 # plot the scores
  4 fig = plt.figure()
  5 ax = fig.add_subplot(111)
  6 plt.plot(np.arange(1, len(scores)+1), scores)
  7 plt.ylabel('Score')
  8 plt.xlabel('Episode #')
  9 plt.show()
 10
 11 plot_scores(scores)
```

```
Episode 16      Average Score: 0.677
Environment solved in -84 episodes!     Average Score: 0.67
```

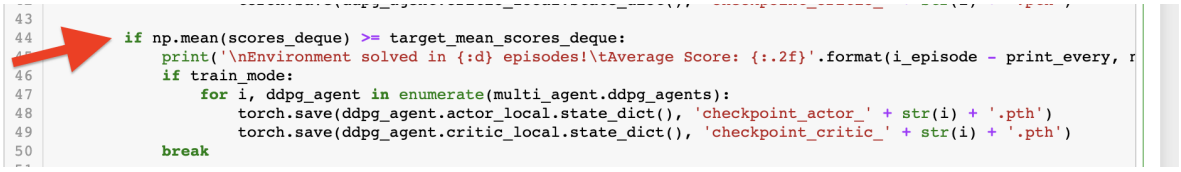




There are no problems in stopping training long after the desired score is reached. Yet, your code is built to stop training once a certain score has been reached. If I input the required score of 0.5, as I did, the training stops before one of the conditions to solve the environment is achieved, that is the average score being calculated over 100 episodes. Also, one of the requirements for the Report is to mention how many episodes it took to solve the environment. The current implementation does not give this information.

So, what to do? *Add a condition that it has been at least 100 episodes before the training is stopped.*

```
39             if train_mode:
40                 for i, ddpg_agent in enumerate(multi_agent.ddpg_agents):
41                     torch.save(ddpg_agent.actor_local.state_dict(), 'checkpoint_actor_' + str(i) + '.pth')
42                     torch.save(ddpg_agent.critic_local.state_dict(), 'checkpoint_critic_' + str(i) + '.pth')
```



```
43
44      if np.mean(scores_deque) >= target_mean_scores_deque:
            print('\nEnvironment solved in {:d} episodes!\tAverage Score: {:.2f}'.format(i_episode - print_every, r
46          if train_mode:
47              for i, ddpg_agent in enumerate(multi_agent.ddpg_agents):
48                  torch.save(ddpg_agent.actor_local.state_dict(), 'checkpoint_actor_' + str(i) + '.pth')
49                  torch.save(ddpg_agent.critic_local.state_dict(), 'checkpoint_critic_' + str(i) + '.pth')
50          break
```

The code is written in PyTorch and Python 3.

The submission includes the saved model weights of the successful agent.

The model weights of the successful agent are saved in:

- `checkpoint_actor_0.pth`,
- `checkpoint_actor_1.pth`,
- `checkpoint_critic_0.pth`, and
- `checkpoint_critic_1.pth`.

README

The GitHub submission includes a `README.md` file in the root of the repository.

The README is submitted in the right format `.md`.

The README describes the the project environment details (i.e., the state and action spaces, and when the environment is considered solved).

The `Introduction` of the README **correctly** describes all the project environment details.

The README has instructions for installing dependencies or downloading needed files.

The README gives correct instructions for:

- ☑ Installing dependencies in the Dependencies section.
- ☑ Downloading the environment in the Getting Started section.

The README describes how to run the code in the repository, to train the agent. For additional resources on creating READMEs or using Markdown, see here and here.

The `Instructions` section of the README **correctly** describes how to run your code.

Report

The submission includes a file in the root of the GitHub repository (one of `Report.md`, `Report.ipynb`, or `Report.pdf`) that provides a description of the implementation.

The Report is concise, comprehensive, and submitted in the required format, `.pdf`.

The report clearly describes the learning algorithm, along with the chosen hyperparameters. It also describes the model architectures for any neural networks.

All requirements are explained.

✅ Learning Algorithm in the `1) Learning algorithm` section.
✅ Hyperparameters in the `3) Parameters used in DDPG algorithm` and `4) Parameters used in DDPG Agent` sections.
✅ Model Architecture in the `2) Network architectures` section.

A plot of rewards per episode is included to illustrate that the agents get an average score of +0.5 (over 100 consecutive episodes, after taking the maximum over both agents).

The submission reports the number of episodes needed to solve the environment.

Great job 👏 Your agents train in a very small number of episodes as demonstrated in the `Plot of Rewards` section.

✅ **The Plot for the trained agents is included.**
❌ **The Report *does not explicitly mention the number of episodes* needed to solve the environment.**

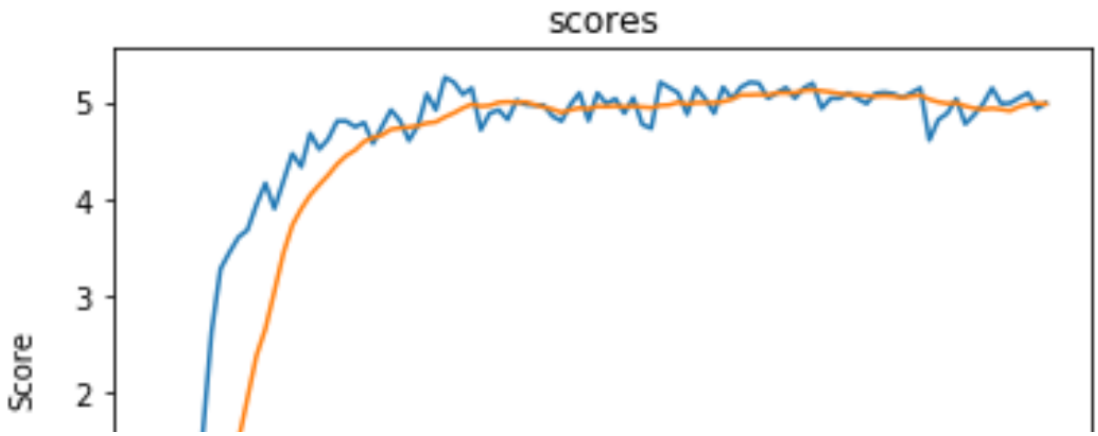
- The requirement here is to explicitly mention how many episodes it took the agent to achieve a score of +0.5. The screenshot:

```
scores = ddpg(target_mean_scores_deque=6.0)

# plot the scores
fig = plt.figure()
ax = fig.add_subplot(111)
plt.plot(np.arange(1, len(scores)+1), scores)
plt.ylabel('Score')
plt.xlabel('Episode #')
plt.show()

plot_scores(scores)
```

```
Episode 100     Average Score: 4.55
Episode 200     Average Score: 5.14
Episode 240     Average Score: 5.15
```



does not offer this information as the agents' score = 4.55 in the first 100 episodes.

Please, print the number of episodes once the condition, average score = 0.5 or higher over 100 episodes, has been satisfied. **Then, add this information to the Report.**

The submission has concrete future ideas for improving the agent's performance.

Good job formulating your arguments. I can see you have a good

Good job formulating your arguments. I can see you have a good understanding of the topic.

- Yes, you can almost always fine-tune the hyperparameters and the model.
- Yes, using different algorithms is an important future step. Here are resources to help with some of your suggestions:
    - A3C
    - PPO and a PPO implementation.
    - D4PG
- Yes, using Prioritized Experience Replay would help a lot. You can also use the more advanced versions, Hindsight Experience Replay and Distributed Prioritized Experience Replay.

I would try building a less complex model. The Tennis environment looks harder to solve than it is. You can do without Batch Normalization and with fewer Fully Connected Layers.

RESUBMIT

DOWNLOAD PROJECT

Learn the best practices for revising and resubmitting your project.

RETURN TO PATH