4180/5180: Reinforcement Learning and Sequential Decision Making (Fall 2022)	Christopher Amato
Northeastern University	Nov. 30th, 2022

Exercise 8: Deep Q-Network (DQN)

Please remember the following policies:

- Exercise due at **11:59 PM EST Nov 30, 2022**.
- Submissions should be made electronically on Canvas. Please ensure that your solutions for both the written and programming parts are present. You can upload multiple files in a single submission, or you can zip them into a single file. You can make as many submissions as you wish, but only the latest one will be considered.
- For **Written** questions, solutions may be handwritten or typeset. If you write your answers by hand and submit images/scans of them, please please ensure legibility and order them correctly in a single PDF file.
- The PDF file should also include the figures from the **Plot** questions.
- For both **Plot** and **Code** questions, submit your source code in Jupyter Notebook (.ipynb file) along with reasonable comments of your implementation. Please make sure the code runs correctly.
- You are welcome to discuss these problems with other students in the class, but you must understand and write up the solution and code yourself. Also, you *must* list the names of all those (if any) with whom you discussed your answers at the top of your PDF solutions page.
- Each exercise may be handed in up to two days late (24-hour period), penalized by 10% per day late. Submissions later than two days will not be accepted.
- Contact the teaching staff if there are medical or other extenuating circumstances that we should be aware of.
- **Notations: RL2e is short for the reinforcement learning book 2nd edition. x.x means the Exercise x.x in the book.**

1. **3 points.** *Nonlinear Function Approximation with Neural Networks*

   Universal approximation theorem tells us Neural Networks have a good deal of approximation capabilities. In this question, we will be approximating a low-dimensional non-linear function $f(x) = 1 + x^2$ using Neural Network through stochastic gradient decent. Specifically, this is a simple regression problem in machine learning. The goal is to learn a parameterized model to predict the value of $f(x)$ given x. To build the neural networks, we will use the PyTorch library (https://pytorch.org), a popular library to build deep neural networks. We recommend you to read the tutorials on the official website if you want to learn more about Pytorch (https://pytorch.org/tutorials/).

   (a) **Code:** We use numpy.linspace to get 500 even spaced values in the range of $[-10, 10]$, complete your training set by obtaining the function outputs for each of the 500 inputs. Complete the *create_dataset* function in the notebook.

   (b) **Code:** Build your two-hidden-layer model with *relu* as activation, layers are of size 8 (dimension). Setup the Adam (Adaptive Moment Estimation) optimizer. Complete the scaffolding code in the notebook.

   (c) **Code:** Randomly sample batches of size 32 from your training set, calculate your loss on the batch (mean squared error), and perform parameter update based on the gradient. We provide you the hyperparameters for training in the notebook. Please complete the related code in the notebook.

   (d) **Plot:** Plot your approximated function and the original function using a dot plot. In each plot, plot the dots for both the predicted value and the ground truth value. Additionally, test and plot your model with layers with dimension = 16, 64 and 128.

   (e) **Written:** How accurate is your learned model? How accurate it is within the range of $[-10, 10]$? How about outside of the range? Can you notice any difference between models having different layer dimensions? What do you think that may have caused the difference?

2. **4 points.** *Four Rooms yet again but with DQN*

   In this question, you will implement DQN and test it on our favorite domain, Four Rooms, as implemented in ex4 and ex7. However, we make one change of the reward function. Specifically, the agent receives -1 for every time step except for reaching the goal location where it receives 0. You will be in charge of your network architecture and other hyperparameters. However, you are recommended to start with the provided hyperparameters in the notebook.

   (a) **Code:** Setup your environment, replay memory buffer, value networks (i.e. the behavior policy network and the target policy network in the notebook), and optimizers. Remember that although your value network models the $Q(s, a)$ value function ($s, a \to q(s, a)$), in the case of DQN it takes the state as input and output vector-form values for all actions ($s \to [q(s, a_1), q(s, a_2), ...]^T$).

   (b) **Code:** Collect rollouts and store them in reply memory while performing batch updates in Q-Learning fashion based on the data in your reply memory. Your loss is the batch MSE of your TD error (the difference between your targets and your value predictions). Also, remember to use ϵ-greedy in your rollouts.

   (c) **Code:** Update your target network (i.e., the target policy network in the notebook) periodically. Repeat the learning process until there is little or no performance gain. Note that, to finish the implementation above, please complete the code in the notebook.

   (d) **Plot, Written:** Plot your learning curve (averaged over 5 trails) with confidence bands.

   (e) **[5180]: 1 point** Implement the double DQN. Compare it with the vanilla one. Comment on your findings.

3. **3 points.** *Evaluate and tune your DQN on classic control tasks*

   In this question, you will evaluate your DQN on CartPole (https://gym.openai.com/envs/CartPole-v1/) and LunarLander (https://gym.openai.com/envs/LunarLander-v2/). You will need to schedule your ϵ so that it is annealed to a low value (e.g. 0.05) from 1.

   (a) **Code:** Run and tune your DQN on CartPole and LunarLander.

   (b) **Plot** Plot your learning curve (averaged over 5 trails) with confidence bands.

   (c) **Written** What are the network architecture and hyperparameters you find works for CartPole and LunarLander respectively?

   (d) **[Extra credit] 1 point Written:** Render your runs at different stages of training, what kind of policy (or policies) do you see? What do you think the corresponding value function would be? You can inspect your value function to validate your intuition.

4. **[Extra credit.] 2 points.** *DQN on Atari*

   (a) **Code:** Evaluate your DQN on any of the Atari game provided with gym (we recommend start with Pong). You will need to add convolution layer. Additionally, you may want to use libraries like OpenAI Baselines for image processing (grayscaling, framestacking, etc.).

   (b) **Plot, Written** Plot your learning curve (averaged over some trails or not). How long does your experiments take to complete? What did you have to do with the hyperparameters to make it work?