Assignment (3)

24789 - Special Topics: Deep Learning for Engineers (Spring 2020)

Author: Yuyang Wang, Amir Barati Farimani
Out Date: 2020/4/4 (Sat)
Due Date: 2020/4/20 (Mon) @ 11:59 pm EST

Submission file structure

/andrewID-HW3
- **/p1**
  - ***.py** (all the Python script files)
  - **p1_model.ckpt**
  - **p1_report.pdf**
- **/p2**
  - ***.py** (all the Python script files)
  - **p2_model.ckpt**
  - **p2_report.pdf**

You can refer to Python3 tutorial, Numpy documentation and PyTorch documentation while working on this assignment. Any deviations from the submission structure shown below would attract penalty to the assignment score. Please use Piazza for any questions on the assignment.

Theory Exercises (50 points)

PROBLEM 1

Graph Convolution [12 points]

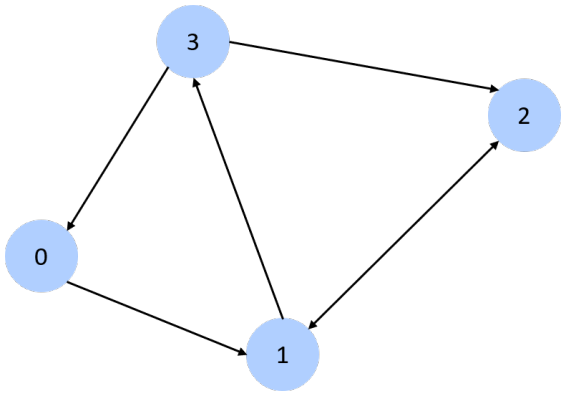


Fig. 1: A directed graph

Consider a simple directed graph shown in Fig. 1. Where the number in each node represents the index, and arrows represent whether two nodes are connected (notice that some connections are unidirectional and some are bidirectional). If there exists an arrow pointing from node A to node B, then node A is connected to node B, but node B is not necessarily connected to node A.

a) [4 points] What is the adjacent matrix, A, of the graph. We use 0 to represent no connection and 1 to represent connection. Also each note should be connected to itself.

b) [4 points] Given the features on each node:

$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ -2 & 0.5 \\ 1 & 3 \\ 0 & -1 \end{bmatrix} \tag{1}$$

where the feature on node i, x_i, is a 2-dimensional vector. Compute new features on each node, x_i', by graph convolutional operation: $X' = \sigma(AX)$, where σ is a ReLU activation function.

c) [4 points] This time, you want to normalize the adjacent matrix by multiplying with the inverse of a degree matrix D. D is a diagonal matrix, where each diagonal element is the summation of each row in A. So the transformed adjacent matrix is given as $A' = D^{-1}A$. Now use A' to compute the new features on each node with $X' = \sigma(A'X) = \sigma(D^{-1}AX)$

PROBLEM 2

Markov Decision Process [12 points]

Given the following MDP as shown in Fig.2, where r represents the reward at each state and numbers on the arrows represent transition probability between different states.

a) [4 points] Which action, A or B, maximizes the expected reward on the following turn starting from State 3?

b) [4 points] Which action, A or B, from State 3, maximizes the total expected discounted future reward, with a discount factor $\gamma = 0.8$? What is the expected discounted future reward for each action?

c) [4 points] For what value of γ does the expected discounted future reward for Action A from State 3 is larger than Action B?

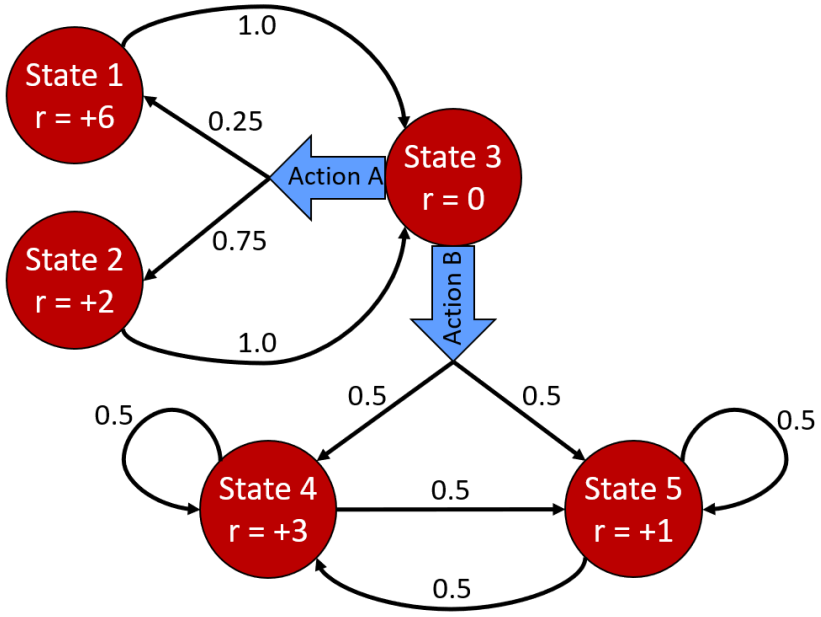


Fig. 2: Markov Decision Process

PROBLEM 3

Q Learning in Reinforcement Learning [16 points]

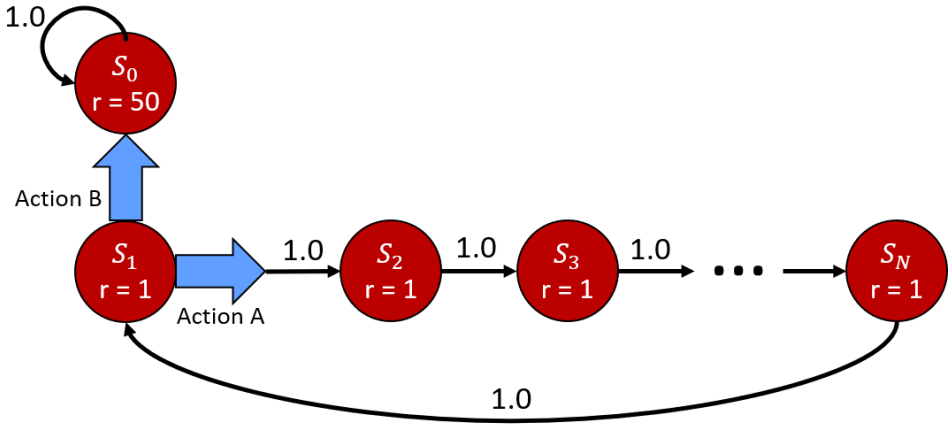


Fig. 3: Reinforcement Learning Model

Consider the MDP in Fig.3. Assume $N > 1000$ and all states except S_1 have only one possible action with a deterministic outcome (transition probability of 1). State S_1 has two possible actions, A and B, each with a deterministic outcome (A always leads to S_2 and B always leads to S_0). Assume a discount factor of $\gamma = 0.5$.

To learn this model we will use Q learning with learning rate $\alpha = 1$. All Q functions for all states are initialized to 0. Whenever we reach state S_1 we use our current Q function estimate to chose the action leading to the highest long term pay. We break ties by choosing action A. We start at state S1.

The update rule for Q learning is given:

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

a) [4 points] After 1 step, what are $Q(S_1, A)$ and $Q(S_1, B)$?

b) [4 points] After 5 steps, what are $Q(S_1, A)$ and $Q(S_1, B)$?

c) [4 points] After $N + 5$ steps, what are $Q(S_1, A)$ and $Q(S_1, B)$?

d) [4 points] When our Q learning converges, what are the convergence values for $Q(S_1, A)$ and $Q(S_1, B)$? What is the convergence value for $Q(S_2, \text{right})$?

PROBLEM 4

Deep Reinforcement Learning [10 points]

Short answer questions:

a) [4 points] What is the difference between on-policy and off-policy reinforcement learning. And name two algorithms which are on-policy and off-policy respectively.

b) [2 points] Why does experience replay help to train the DQN model?

c) [2 points] What is the difference between Q-learning and policy gradients methods?

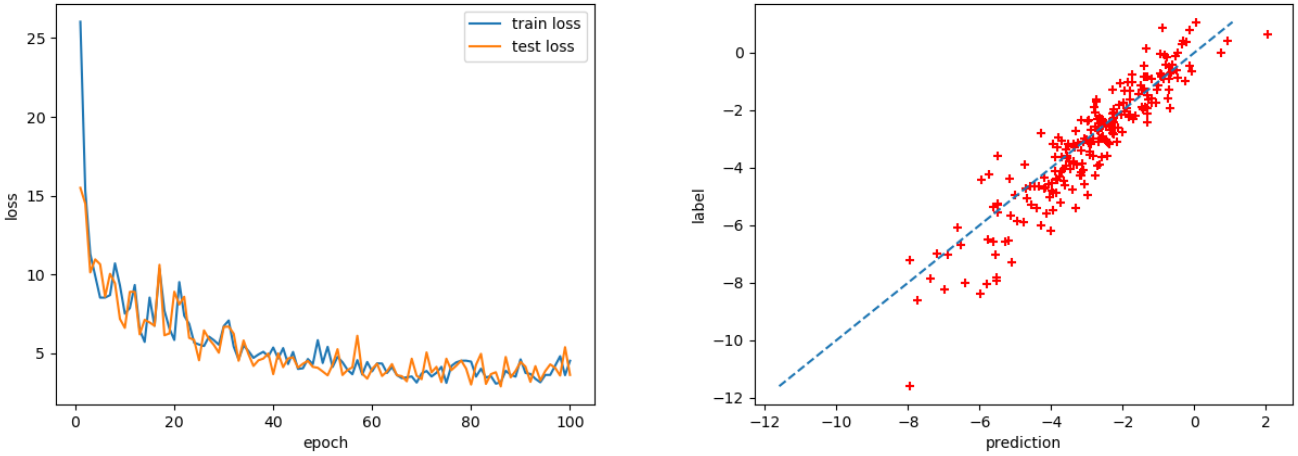
d) [2 points] What is the role of the critic in actor-critic algorithm? And why the critic can help learn the optimal policy?

Programming Exercises (50 points)

PROBLEM 1

Graph Neural Network (25 points)

In this question, you are asked to implement a graph neural networks based upon PyTorch and PyTorch Geometric to predict the aqueous solubility from molecular structure.



(a) Loss vs. training epoch

(b) Prediction vs. label

Fig. 4: Sample plots using GCN

PyTorch Geometric (PyG) is a geometric deep learning extension library for PyTorch, which consists of various methods for deep learning on graphs and other irregular structures. Since cuda support for PyG varies on different platforms, we recommend you install CPU version of the package. Notice that cuda support of your PyG should be consistent with PyTorch, so you may want to create a new virtual environment with CPU version pytorch if you used to have cuda supported PyTorch installed. The environment setting is given:

```
$ conda create -n your_env_name python=3.6
$ conda activate your_env_name
$ conda install -c conda-forge matplotlib
$ conda install -c anaconda pandas
$ conda install -c conda-forge rdkit
$ conda install pytorch torchvision cpuonly -c pytorch
$ pip install torch-scatter==latest+cpu -f https://pytorch-geometric.com/whl/torch-1.4.0.html
$ pip install torch-sparse==latest+cpu -f https://pytorch-geometric.com/whl/torch-1.4.0.html
$ pip install torch-cluster==latest+cpu -f https://pytorch-geometric.com/whl/torch-1.4.0.html
$ pip install torch-spline-conv==latest+cpu -f https://pytorch-geometric.com/whl/torch-1.4.0.html
$ pip install torch-geometric
```

To check if torch geometric is correctly installed, you can download the example code from their github:

```
$ git clone https://github.com/rusty1s/pytorch_geometric.git
$ cd examples
$ python gcn.py
```

The problem you will be working on is a regression problem, where given the molecular structures, you should build a graph neural network to predict the log solubility.

A template code has been given for you to start, but you are not required to follow the code. The `get_dataset(PATH)` function loads graph-featurized training set and test set. For each sample in the dataset, it contains node features, log solubility, edge index, and edge attributes. You can refer to the example code from PyG to build and train your own model: gcn.py. We recommend you to use `torch_geometric.nn.GCNConv()` which implements the graph convolutional operator from Semi-supervised Classification with Graph Convolutional Networks and `torch_geometric.nn.ChebConv()` which implements the chebyshev spectral graph convolutional operator from Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. And you can find more information in PyG documentation.

In your submission to Gradescope, you should submit all your codes and your model checkpoints. Also, a report should be included concerning your model, hyperparameters, performance on test set, and plot of your training process as shown in Fig. 4. To get full credits, the summation of mean square error between model prediction and label on **test set** should be smaller than 200. You will get 5 bonus credits if the MSE on test set is smaller than 150.

PROBLEM 2

DQN in OpenAI Gym (25 points)

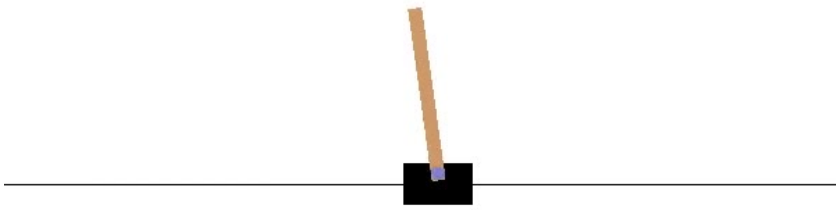


Fig. 5: CartPole-v1

In this question, you are asked to implement a DQN model (Deep Q-Network) based upon PyTorch to solve the cart pole problem in OpenAI gym. OpenAI gym is a toolkit for developing and comparing reinforcement learning algorithms. This is the gym open-source library, which gives you access to a standardized set of environments.

You can follow the instructions below to install OpenAI gym on your anaconda. Notice that here we only need a minimal install of the package. If you haven't used gym before, we recommend you to install the package directly from PyPI by `pip install gym`.

```
$ conda create -n your_env_name
$ conda activate your_env_name
$ pip install gym
```

The environment you will try to solve is CartPole-v1 as shown in Fig. 5, where the system is controlled by applying a force of +1 or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every timestep that the pole remains upright. The episode ends when the pole is more than 15 degrees from vertical.

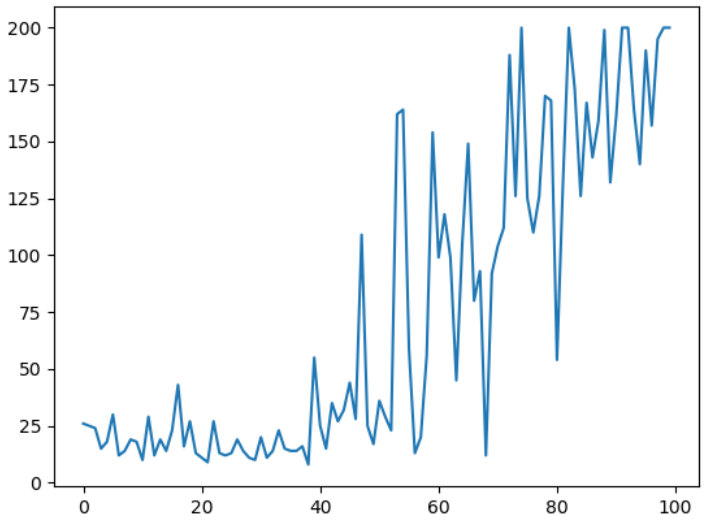


Fig. 6: Duration vs. episode

The original paper on DQN can be found here: Playing Atari with Deep Reinforcement Learning and Human-level control through deep reinforcement learning. And you can find the useful PyTorch tutorial: Reinforcement Learning (DQN) Tutorial. Notice that in the tutorial, image of each time step is used as input, however in this assignment, we only require you to use the 4-dimensional state from wrapped CartPole-v0 environment to train the DQN model. A template code has been given for you to start, but you are not required to follow the code.

In your submission to Gradescope, you should submit all your codes and your model checkpoints. Also, a report should be included concerning your model, hyperparameters, performance on 10 test episodes, and plot of your training process as shown in Fig. 6. A video recorded when testing the DQN model should be submitted as well. A sample video can be found: Open.AI Cartpole-v1 - Neural Network Solution. To get full credits, the mean duration of all the 10 episodes during test with your DQN agent should be greater than 200. Moreover, you will get 15 bonus points if you implement double-DQN (reference paper can be found here: Deep Reinforcement Learning with Double Q-learning) besides the vanilla DQN algorithm and submit all the codes, report and videos about double-DQN to Gradescope.