Q-Learning & DQNs (30 points + 5 bonus points)

In this section, we will implement a few key parts of the Q-Learning algorithm for two cases - (1) A Q-network which is a single linear layer (referred to in RL literature as "Q-learning with linear function approximation") and (2) A deep (convolutional) Q-network, for some Atari game environments where the states are images.

Optional Readings:

- **Playing Atari with Deep Reinforcement Learning**, Mnih et. al., https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf (https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf)
- **The PyTorch DQN Tutorial** https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html (https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html)

Note: **The bonus credit for this question applies to both sections CS 7643 and CS 4803**

In [0]:

```
%load_ext autoreload
%autoreload 2

import numpy as np
import gym

import torch
import torch.nn as nn
import torch.optim as optim

from core.dqn_train import DQNTrain
from utils.test_env import EnvTest
from utils.schedule import LinearExploration, LinearSchedule
from utils.preprocess import greyscale
from utils.wrappers import PreproWrapper, MaxAndSkipEnv

from linear_qnet import LinearQNet
from cnn_qnet import ConvQNet
from my_cnn_qnet import ConvQNet as MyConvQNet

if torch.cuda.is_available():
    device = torch.device('cuda', 0)
else:
    device = torch.device('cpu')
```

In [6]:

```
device
```

Out[6]:

```
device(type='cuda', index=0)
```

Deliverable 1 (15 points)

Run the following block of code to train a Linear Q-Network. You should get an average reward of ~4.0, full credit will be given if average reward at the final evaluation is above 3.5

In [0]:

```
from configs.p1_linear import config as config_lin

env = EnvTest((5, 5, 1))

# torch.manual_seed(0)
# np.random.seed(0)

# exploration strategy
exp_schedule = LinearExploration(env, config_lin.eps_begin,
        config_lin.eps_end, config_lin.eps_nsteps)

# learning rate schedule
lr_schedule  = LinearSchedule(config_lin.lr_begin, config_lin.lr_end,
        config_lin.lr_nsteps)

# train model
model = DQNTrain(LinearQNet, env, config_lin, device)
model.run(exp_schedule, lr_schedule)
```

```
Evaluating...
Average reward: 0.50 +/- 0.00

 1001/10000 [==>...........................] - ETA: 10s - Loss: 0.2570 - Avg_R: 0.7600 - Max_R:
2.3000 - eps: 0.8020 - Grads: 0.6206 - Max_Q: 0.8789 - lr: 0.0042

Evaluating...
Average reward: 3.80 +/- 0.00

 2001/10000 [=====>........................] - ETA: 9s - Loss: 0.4668 - Avg_R: 1.6350 - Max_R:
3.9000 - eps: 0.6040 - Grads: 0.8322 - Max_Q: 1.9482 - lr: 0.0034

Evaluating...
Average reward: 4.10 +/- 0.00

 3001/10000 [========>.....................] - ETA: 8s - Loss: 0.3156 - Avg_R: 2.2450 - Max_R:
4.1000 - eps: 0.4060 - Grads: 0.6215 - Max_Q: 2.4127 - lr: 0.0026

Evaluating...
Average reward: 3.80 +/- 0.00

 4001/10000 [===========>..................] - ETA: 7s - Loss: 0.1326 - Avg_R: 2.8950 - Max_R:
4.1000 - eps: 0.2080 - Grads: 0.5427 - Max_Q: 2.6305 - lr: 0.0018

Evaluating...
Average reward: 4.10 +/- 0.00

 5001/10000 [==============>...............] - ETA: 6s - Loss: 0.1431 - Avg_R: 3.8500 - Max_R:
4.1000 - eps: 0.0100 - Grads: 0.4613 - Max_Q: 2.7319 - lr: 0.0010

Evaluating...
Average reward: 4.10 +/- 0.00

 6001/10000 [=================>............] - ETA: 5s - Loss: 0.0283 - Avg_R: 3.9500 - Max_R:
4.1000 - eps: 0.0100 - Grads: 0.2279 - Max_Q: 2.6787 - lr: 0.0010

Evaluating...
Average reward: 4.10 +/- 0.00

 7001/10000 [====================>.........] - ETA: 3s - Loss: 0.0122 - Avg_R: 4.0050 - Max_R:
4.1000 - eps: 0.0100 - Grads: 0.1966 - Max_Q: 2.6001 - lr: 0.0010

Evaluating...
Average reward: 4.10 +/- 0.00

 8001/10000 [=======================>......] - ETA: 2s - Loss: 0.1128 - Avg_R: 4.0850 - Max_R:
4.1000 - eps: 0.0100 - Grads: 0.1390 - Max_Q: 2.8187 - lr: 0.0010

Evaluating...


Average reward: 4.10 +/- 0.00

 9001/10000 [==========================>...] - ETA: 1s - Loss: 0.0002 - Avg_R: 3.9000 - Max_R:
4.1000 - eps: 0.0100 - Grads: 0.0297 - Max_Q: 2.6170 - lr: 0.0010

Evaluating...


Average reward: 4.10 +/- 0.00

10001/10000 [==============================] - 12s - Loss: 0.0066 - Avg_R: 4.0850 - Max_R: 4.10
00 - eps: 0.0100 - Grads: 0.2125 - Max_Q: 2.5012 - lr: 0.0010

- Training done.
Evaluating...
Average reward: 4.10 +/- 0.00
```

Deliverable 2

In [0]:

```python
from configs.p2_cnn import config as config_cnn

env = EnvTest((80, 80, 1))

# exploration strategy
exp_schedule = LinearExploration(env, config_cnn.eps_begin,
        config_cnn.eps_end, config_cnn.eps_nsteps)

# learning rate schedule
lr_schedule  = LinearSchedule(config_cnn.lr_begin, config_cnn.lr_end,
        config_cnn.lr_nsteps)

# train model
model = DQNTrain(MyConvQNet, env, config_cnn, device)
model.run(exp_schedule, lr_schedule)
```

```
Evaluating...
Average reward: 0.50 +/- 0.00

Populating the memory 150/200...

Evaluating...


Average reward: -0.70 +/- 0.00

 301/1000 [========>.....................] - ETA: 1s - Loss: 0.0459 - Avg_R: 0.1700 - Max_R: 2.
2000 - eps: 0.4060 - Grads: 1.7887 - Max_Q: 0.1710 - lr: 0.0002

Evaluating...
Average reward: 0.50 +/- 0.00

 401/1000 [===========>.................] - ETA: 1s - Loss: 0.0234 - Avg_R: 0.3450 - Max_R: 2.
3000 - eps: 0.2080 - Grads: 0.4504 - Max_Q: 0.2027 - lr: 0.0001

Evaluating...
Average reward: 0.50 +/- 0.00

 501/1000 [==============>..............] - ETA: 1s - Loss: 0.0030 - Avg_R: 0.4750 - Max_R: 2.
3000 - eps: 0.0100 - Grads: 0.2430 - Max_Q: 0.2037 - lr: 0.0001

Evaluating...
Average reward: 0.50 +/- 0.00

 601/1000 [=================>...........] - ETA: 1s - Loss: 0.2259 - Avg_R: 2.2400 - Max_R: 4.
0000 - eps: 0.0100 - Grads: 3.0440 - Max_Q: 0.2658 - lr: 0.0001

Evaluating...
Average reward: 4.00 +/- 0.00

 701/1000 [====================>.........] - ETA: 1s - Loss: 0.3333 - Avg_R: 3.7200 - Max_R: 4.
0000 - eps: 0.0100 - Grads: 4.0482 - Max_Q: 0.3772 - lr: 0.0001

Evaluating...


Average reward: 1.60 +/- 0.00

 801/1000 [=======================>......] - ETA: 0s - Loss: 0.0598 - Avg_R: 3.0100 - Max_R: 4.
0000 - eps: 0.0100 - Grads: 4.3118 - Max_Q: 0.4821 - lr: 0.0001

Evaluating...


Average reward: 4.00 +/- 0.00

 901/1000 [==========================>...] - ETA: 0s - Loss: 0.0193 - Avg_R: 3.7200 - Max_R: 4.
1000 - eps: 0.0100 - Grads: 1.0536 - Max_Q: 0.5751 - lr: 0.0001

Evaluating...


Average reward: 4.10 +/- 0.00

1001/1000 [==============================] - 4s - Loss: 0.0022 - Avg_R: 4.0950 - Max_R: 4.1000
- eps: 0.0100 - Grads: 0.9566 - Max_Q: 0.6519 - lr: 0.0001

- Training done.
Evaluating...
Average reward: 4.10 +/- 0.00
```

You should get a final average reward of over 4.0 on the test environment, similar to the previous case.

Part 3: Playing Atari Games from Pixels - using Linear Function Approximation

Now that we have setup our Q-Learning algorithm and tested it on a simple test environment, we will shift to a harder environment - an Atari 2600 game from OpenAI Gym: Pong-v0 (https://gym.openai.com/envs/Pong-v0/ (https://gym.openai.com/envs/Pong-v0/)), where we will use RGB images of the game screen as our observations for state.

No additional implementation is required for this part, just run the block of code below (will take around 1 hour to train). We don't expect a simple linear Q-network to do well on such a hard environment - full credit will be given simply for running the training to completion irrespective of the final average reward obtained.

You may edit `configs/p3_train_atari_linear.py` if you wish to play around with hyperparamters for improving performance of the linear Q-network on Pong-v0, or try another Atari environment by changing the `env_name` hyperparameter. The list of all Gym Atari environments are available here: https://gym.openai.com/envs/#atari (https://gym.openai.com/envs/#atari)

Deliverable 3 (5 points)

Run the following block of code to train a linear Q-network on Atari Pong-v0. We don't expect the linear Q-Network to learn anything meaingful so full credit will be given for simply running this training to completion (without errors), irrespective of the final average reward.

In [0]:

```
from configs.p3_train_atari_linear import config as config_lina

# make env
env = gym.make(config_lina.env_name)
env = MaxAndSkipEnv(env, skip=config_lina.skip_frame)
env = PreproWrapper(env, prepro=greyscale, shape=(80, 80, 1),
                    overwrite_render=config_lina.overwrite_render)

# exploration strategy
exp_schedule = LinearExploration(env, config_lina.eps_begin,
        config_lina.eps_end, config_lina.eps_nsteps)

# learning rate schedule
lr_schedule  = LinearSchedule(config_lina.lr_begin, config_lina.lr_end,
        config_lina.lr_nsteps)

# train model
model = DQNTrain(LinearQNet, env, config_lina, device)
print("Linear Q-Net Architecture:\n", model.q_net)
model.run(exp_schedule, lr_schedule)
```

```
Evaluating...

Linear Q-Net Architecture:
 LinearQNet(
  (linear_layer): Linear(in_features=25600, out_features=6, bias=True)
)

Average reward: -21.00 +/- 0.00

250201/500000 [==============>...............] - ETA: 1382s - Loss: 0.0602 - Avg_R: -20.5000 -
Max_R: -18.0000 - eps: 0.7748 - Grads: 5.5457 - Max_Q: 11.6656 - lr: 0.0001

Evaluating...


Average reward: -21.00 +/- 0.00

500001/500000 [==============================] - 2927s - Loss: 0.4860 - Avg_R: -20.4200 - Max_R
: -17.0000 - eps: 0.5500 - Grads: 44.0041 - Max_Q: 11.4117 - lr: 0.0001

- Training done.
Evaluating...


Average reward: -20.98 +/- 0.02
```

Part 4: [BONUS] Playing Atari Games from Pixels - using Deep Q-Networks

This part is extra credit and worth 5 bonus points. We will now train our deep Q-Network from Part 2 on Pong-v0.

Again, no additional implementation is required but you may wish to tweak your CNN architecture in `cnn_qnet.py` and hyperparameters in `configs/p4_train_atari_cnn.py` (however, evaluation will be considered at no farther than the default 5 million steps, so you are not allowed to train for longer). Please note that this training may take a very long time (we tested this on a single GPU and it took around 6 hours).

The bonus points for this question will be allotted based on the best evaluation average reward (EAR) before 5 million time stpes:

1. EAR >= 0.0 : 4/4 points
2. EAR >= -5.0 : 3/4 points
3. EAR >= -10.0 : 3/4 points
4. EAR >= -15.0 : 1/4 points

Deliverable 4: (5 bonus points)

Run the following block of code to train your DQN:

In [0]:

```
## Tweaks
```

In [0]:

```
from configs.p4_train_atari_cnn import config as config_cnna


# make env
env = gym.make(config_cnna.env_name)
env = MaxAndSkipEnv(env, skip=config_cnna.skip_frame)
env = PreproWrapper(env, prepro=greyscale, shape=(80, 80, 1),
                    overwrite_render=config_cnna.overwrite_render)

# exploration strategy
exp_schedule = LinearExploration(env, config_cnna.eps_begin,
        config_cnna.eps_end, config_cnna.eps_nsteps)

# learning rate schedule
lr_schedule  = LinearSchedule(config_cnna.lr_begin, config_cnna.lr_end,
        config_cnna.lr_nsteps)

# train model
model = DQNTrain(MyConvQNet, env, config_cnna, device)
print("CNN Q-Net Architecture:\n", model.q_net)
model.run(exp_schedule, lr_schedule)
```

```
Evaluating...

CNN Q-Net Architecture:
 ConvQNet(
  (first_layer): Conv2d(4, 8, kernel_size=(4, 4), stride=(2, 2))
  (relu1): ReLU()
  (second_layer): Conv2d(8, 16, kernel_size=(4, 4), stride=(2, 2))
  (relu2): ReLU()
  (third_layer): Conv2d(16, 32, kernel_size=(4, 4), stride=(2, 2))
  (relu3): ReLU()
  (linear_layer1): Linear(in_features=2048, out_features=512, bias=True)
  (relu4): ReLU()
  (linear_layer2): Linear(in_features=512, out_features=6, bias=True)
)

Average reward: -20.70 +/- 0.10

 250101/5000000 [>............................] - ETA: 29194s - Loss: 0.0239 - Avg_R: -19.8600
- Max_R: -17.0000 - eps: 0.7749 - Grads: 0.2247 - Max_Q: 0.0287 - lr: 0.0003

Evaluating...


Average reward: -17.32 +/- 0.22

 500701/5000000 [==>..........................] - ETA: 30117s - Loss: 0.0290 - Avg_R: -18.9000
- Max_R: -15.0000 - eps: 0.5494 - Grads: 0.3455 - Max_Q: 0.5849 - lr: 0.0003
```

```
Evaluating...

Average reward: -13.56 +/- 0.46

 750801/5000000 [===>..........................] - ETA: 29279s - Loss: 0.0260 - Avg_R: -16.9800
- Max_R: -11.0000 - eps: 0.3243 - Grads: 0.2333 - Max_Q: 0.9832 - lr: 0.0003

Evaluating...

Average reward: -12.42 +/- 0.43

1001601/5000000 [=====>........................] - ETA: 28022s - Loss: 0.0345 - Avg_R: -11.7400
- Max_R: -3.0000 - eps: 0.1000 - Grads: 0.3950 - Max_Q: 1.3342 - lr: 0.0002

Evaluating...

Average reward: -8.52 +/- 0.65

1252201/5000000 [======>.......................] - ETA: 26634s - Loss: 0.0268 - Avg_R: -8.3400
- Max_R: 3.0000 - eps: 0.1000 - Grads: 0.3940 - Max_Q: 1.5062 - lr: 0.0002

Evaluating...

Average reward: -5.00 +/- 0.92

1502401/5000000 [========>.....................] - ETA: 25170s - Loss: 0.0523 - Avg_R: -5.6000
- Max_R: 12.0000 - eps: 0.1000 - Grads: 0.4431 - Max_Q: 1.6281 - lr: 0.0002

Evaluating...

Average reward: -2.76 +/- 0.99

1753201/5000000 [=========>....................] - ETA: 23662s - Loss: 0.0196 - Avg_R: -5.8200
- Max_R: 7.0000 - eps: 0.1000 - Grads: 0.4053 - Max_Q: 1.6638 - lr: 0.0002

Evaluating...

Average reward: -1.92 +/- 1.03

2003501/5000000 [===========>..................] - ETA: 22016s - Loss: 0.0326 - Avg_R: -6.7600
- Max_R: 10.0000 - eps: 0.1000 - Grads: 0.4430 - Max_Q: 1.5805 - lr: 0.0002

Evaluating...

Average reward: -1.28 +/- 0.83

2254401/5000000 [============>.................] - ETA: 20333s - Loss: 0.0114 - Avg_R: -5.9800
- Max_R: 7.0000 - eps: 0.1000 - Grads: 0.2884 - Max_Q: 1.6442 - lr: 0.0002

Evaluating...

Average reward: -3.46 +/- 0.80

2504501/5000000 [==============>...............] - ETA: 18615s - Loss: 0.0700 - Avg_R: -4.1000
- Max_R: 13.0000 - eps: 0.1000 - Grads: 0.8565 - Max_Q: 1.7238 - lr: 0.0001

Evaluating...

Average reward: -2.40 +/- 0.80

2620001/5000000 [==============>...............] - ETA: 17885s - Loss: 0.0245 - Avg_R: -6.1200
- Max_R: 10.0000 - eps: 0.1000 - Grads: 0.4907 - Max_Q: 1.5983 - lr: 0.0001
```

In [0]:

```

```