Project: Navigation

Student: Ali Hummos

Date: 7/5/19

Overview:

This the first project of the Udacity deep RL nanodegree and it aims to solve the Unity Banana environment, achieving a score of >13 on 100 consecutive episodes.

This project used a DQN model to solve the environment. Through Bayesian Hyperparameter optimization, the model was able to solve the environment in 192 episodes.

Implementation:

The code submitted uses a DQN agent, in a model-free RL framework. The agents learns through interacting with the environment and storing an experience replay buffer as is described in the original DQN nature paper.

This value-based learning uses an MLP neural network to estimate the action value in the current state. The agent observes any rewards received, and observes the next state it lands in. It attempts to update its beliefs about the value of the action it just took, by considering the reward received and an estimate of the value of the state it landed in. A separate set of weights, modelling a target network, provides an estimate of the value of this future state. The target network is identical in structure but is updated less often than the active Q-network.

The structure of the neural network used is as follows:

- 37 relu units as input layer
- 64 relu units as hidden layer 1
- 64 relu units as hidden layer 2
- 4 units as output layer with no activation function.
-

Hyperparameters:

There are several hyperparameters that guide the agent's behavior and learning algorithm.

- BUFFER_SIZE = int(1e5) # replay buffer size
- BATCH_SIZE = 64 # minibatch size
- GAMMA = 0.99 # discount factor
- TAU = 1e-3 # for soft update of target parameters
- LR = 5e-4 # learning rate
- UPDATE_EVERY = 4 # how often to update the network
- EPS_DECAY #Decay rate of Epsilon

I empirically chose three parameters to optimize through Bayesian Optimization (using python package Bayesian-optimization). After optimizing 'BUFFER_SIZE' 'TAU' and 'EPS_DECAY', the algorithm reached at these optimal values, respectively:

```
buffer... | eps_decay |    tau     |
 3.578e+5 |  0.8058   |  0.003205 |
```

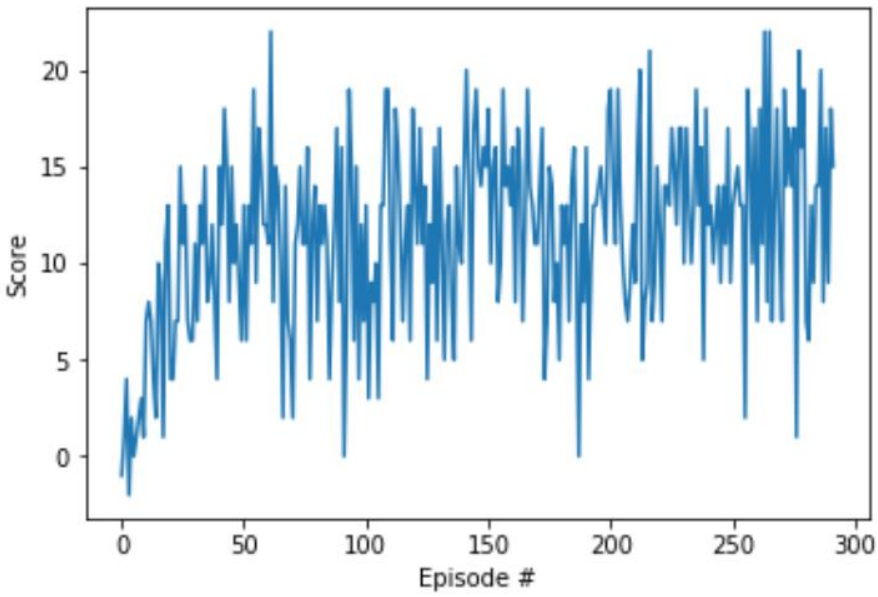
Training:

The agent was trained locally on a NVIDIA TITAN RTX graphics card. Parameter optimization was implemented through the pypi package 'bayesian-optimization'. Training a single DQN network requires about 20 minutes. Bayesian optimization took about 3 hours of GPU time.

Solved in: Environment was solved in 113 episodes in some experiments.

Plot: Here I provide a plot of one set of parameters that was successful in solving the environment in 192 episodes.

```
Episode 100     Average Score: 9.31
Episode 200     Average Score: 12.06
Episode 292     Average Score: 13.05
Environment solved in 192 episodes!     Average Score: 13.05
```



Future Work:

This work presents a DQN algorithm, with some of the parameters optimized through Bayesian Optimization. Several extensions to the DQN algorithm has since been contributed and implementing some of these would boost the performance of this agent [1, 2]. Other methods also have recently been

researched to train DQN by neuroevolution rather than gradient descent [3] and might provide more stability if combined with gradient methods. In addition, a more extensive parameters search is still in order, as this code only explores only a subset of the hyperparameters.

References:

1) Anschel, O., Baram, N., & Shimkin, N. (2017, August). Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (pp. 176-185). JMLR. org.

2) Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., ... & Silver, D. (2018, April). Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

3) Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., & Clune, J. (2017). Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06*