3547 Assignment 4

This assignment uses Deep Q Learning to enable you to build an agent that will learn to play WumpusWorld from experience interacting with your Environment code from Assignment 1. The algorithm we will use is DQN with a replay buffer and target network. The associated code example is a working implementation that you can modify in any of a number of ways.

Notes

1) This assignment uses a new agent called DeepQAgent. The agent's belief state will need to include at least the following features[1]:
   a) A 4 x 4 array with 1 where the agent is (0 everywhere else)
   b) A set of 4 indicator variables representing the agent's orientation
   c) A 4 x 4 array with 1 everywhere the agent has visited (else 0)
   d) A 4 x 4 array with 1 everywhere a Stench has been detected (else 0)
   e) A 4 x 4 array with 1 everywhere a Breeze has been detected (else 0)
   f) A Boolean which is True if the agent has the Gold
   g) A Boolean which is True if the agent perceives a Glitter
   h) A Boolean which is True is the agent has the arrow (i.e. has not used it)
   i) A Boolean which is True if the agent has ever heard a Scream
2) The example includes additional features that are intended to reduce how often the agent gets caught in a loop in the state graph. With only the features above the agent could, for example, turn right 4 times in a row and end up in the same state as it started in, in which case it would loop forever unless the loop is broken somehow. We will discuss this in the webinars.
3) You may want to experiment with the size of the net (depth and width). The size of the first layer will need to match the size of the input (4 * 16 + 4 + 4 = 72 if you used just the inputs as per item 1 above) and the size of the output should be 6 to produce a Q value over the 6 possible actions.
4) The example uses an epsilon-greedy policy (as it should) to select the action during training.

Requirements for this Assignment

Try another scheme other than the one used in the example to try to avoid loops in the graph. Does it perform better or worse than the example?

Extensions for the Project

If you would like to do an extension of Assignment 4 as your project, here are some ideas:

1) Add a 3x3 convolutional input and train the net to play on grids that are larger than 4x4.
2) Try different depths/widths of neural nets and compare their performance.
3) Use your ProbAgent as a Critic to train an Actor-Critic agent. Compare the training time to the DQNAgent's.

[1] or equivalent representations that can be mapped to these features for input to the neural net

4) Try removing the feature that keeps track of whether the agent has the arrow available or not and see how much it degrades the agent's performance.
5) Try implementing a different algorithm such as a Policy Gradient method.
6) Develop a way of visualizing what the agent has learned.