Q-learning is a model-free reinforcement learning algorithm to learn the value of an action in a particular state. It does not require a model of the environment (hence "model-free"), and it can handle problems with stochastic transitions and rewards without requiring adaptations.

For any finite Markov decision process (FMDP), Q-learning finds an optimal policy in the sense of maximizing the expected value of the total reward over any and all successive steps, starting from the current state.[1] Q-learning can identify an optimal action-selection policy for any given FMDP, given infinite exploration time and a partly random policy. "Q" refers to the function that the algorithm computes – the expected rewards for an action taken in a given state. Q-learning seeks to find the best action to take given its current state. More specifically, q-learning seeks to learn a policy that maximizes the total reward.

My QLearner class consists of two methods and a constructor. The constructor sets the number of states equal to one hundred, the number of actions equal to four, the random action rate, which is the probability of selecting a random action at each step and should range between 0.0 (no random actions) to 1.0 (always random action) with 0.5 as a typical value. The random action decay rate, after each update, rar = rar \* radr. Ranges between 0.0 (immediate decay to 0) and 1.0 (no decay) and is typically 0.99. It also sets the dyna, which is the number of dyna updates for each regular update. When Dyna is used, 200 is a typical value. It also sets the alpha value, which is the learning rate used in the update rule. Should range between 0.0 and 1.0 with 0.2 as a typical value. It finally sets the gamma, which is discount rate used in the update rule. Should range between 0.0 and 1.0 with 0.9 as a typical value. The constructor also sets the Q table to its initial setting and initializes the dyna per the lecture.The first method is querySetState. This method takes in the new state as its only parameter and returns the selected action. The second method is query which takes in the new state and the immediate reward as parameters and returns the selected action.

I analyzed alpha and gamma using brute force of testing using increments of 0.1 for each and determining which pair of values returned the highest reward for each of the three worlds. For world 4, the highest median reward I got was –4759 using an alpha of 0.6 and a gamma of 0.6. When using a gamma of 0, I got a reward of -10,000 every time. For world 4 it seems the higher the gamma, the lower the reward. It also seems like for some instances, higher alphas lead to lower rewards. For world 5, every single alpha and gamma value that I used had a median reward of -10,000. This may have been due to the fact that the board was very scattered and random, and therefore there were no patterns to learn from. For world 1, the highest median reward I got was -4403.5, using an alpha of 0.1 and a gamma of 0.9. When using a gamma of 0, I got a reward of -10,000 every time. For world 1 it seems the higher the gamma, the lower the reward.