

**Comprehension - Training pseudo code**

The deep Q-learning pseudocode is as follows:

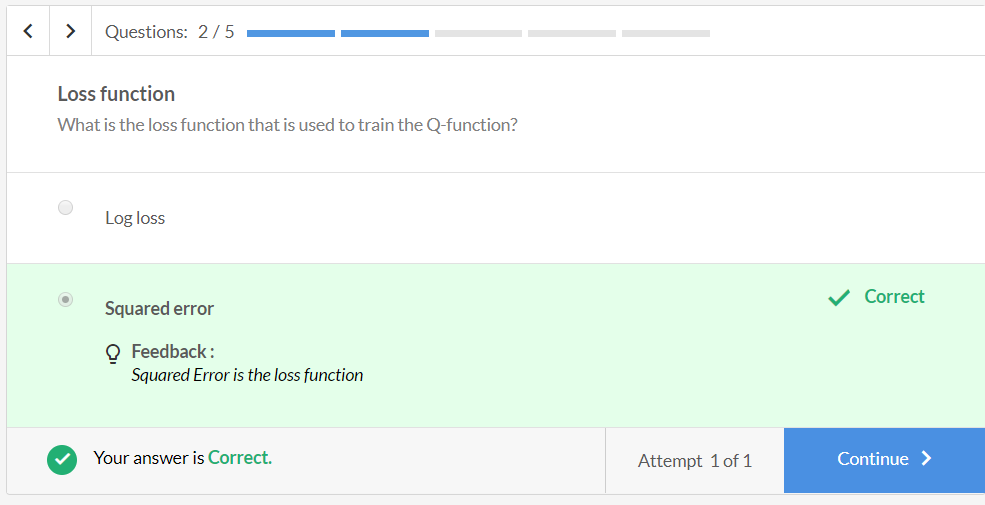
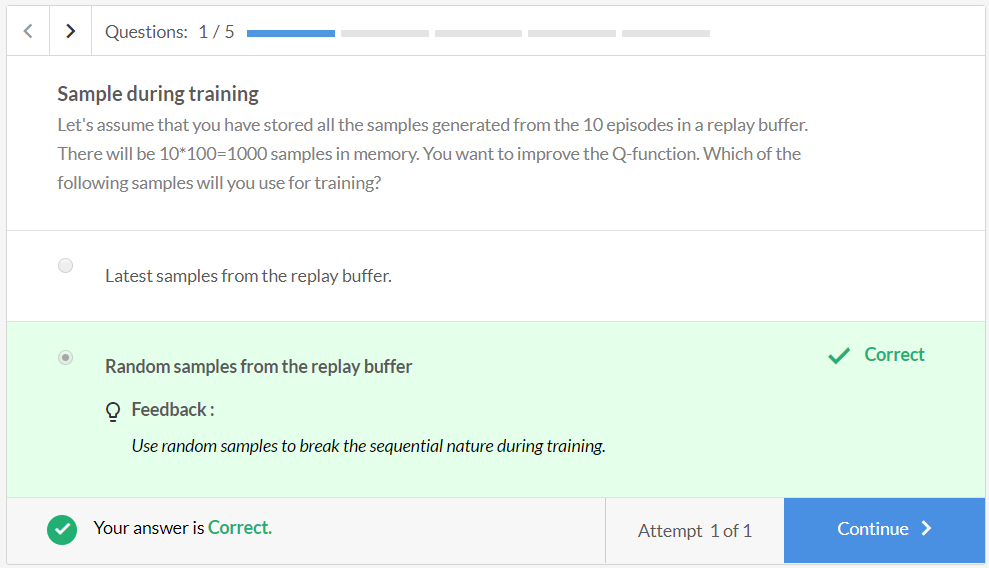
* Initialise replay memory D to a capacity of N (if N=2000, then it can store 2000 experiences)
* Initialise the action-value function Q (i.e. the neural net) with random weights
* Total number of episodes is M
* Each episode is of length (time steps) T
* For 1 to M episodes, do:
  + For 1 to T time steps, do:
    - **Generate an experience** of the form <s,a,s′,r>
      * With probability epsilon, select a random action **a**
      * Else selecta=arg maxaQ(s′,a,θ)
      * Go to the next state s'
      * Set next state as the current state
      * Store the experience in the replay memory D
    - **Train the model** on (say) 100 samples (batch size) randomly selected from the memory
      * Randomly sample transitions (s,a,s′,r) of batch size from replay buffer
      * Calculate the target (y): r+maxaQ(s′,a)
      * Calculate the Q-value for this state-action pair (s,a) as predicted by the network
      * Train the model to minimize the 'squared error':

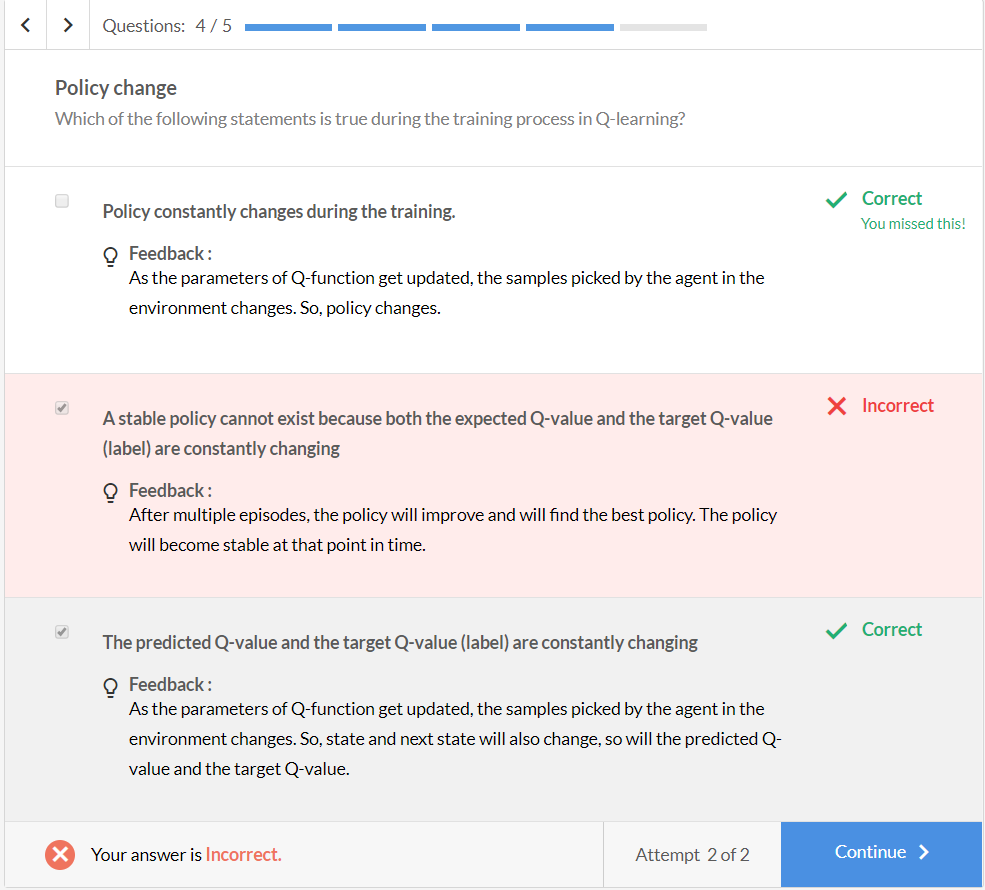
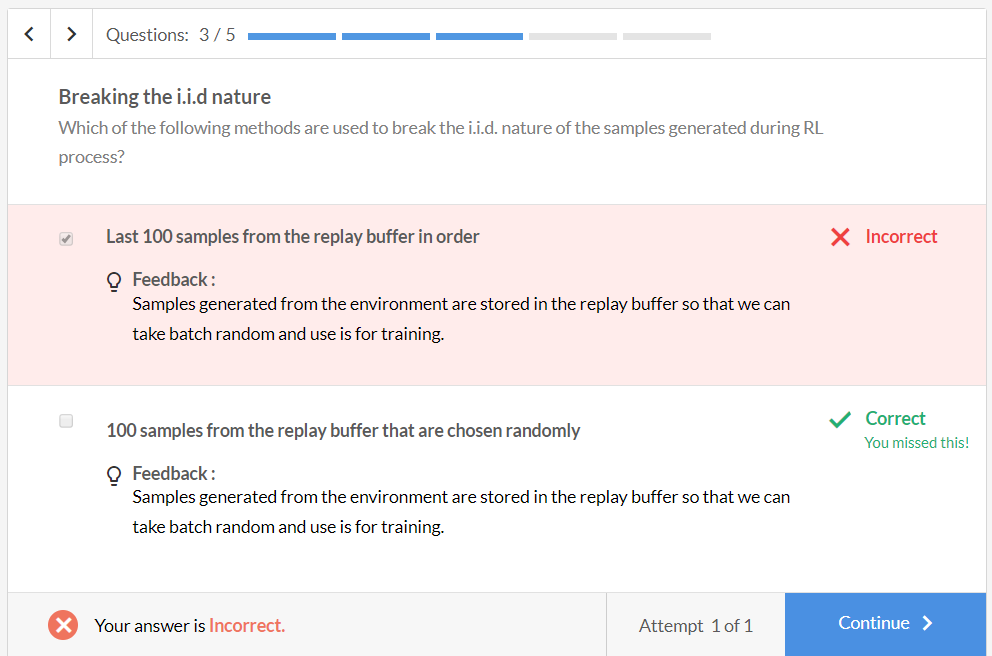
(Q(s, a)−y)2

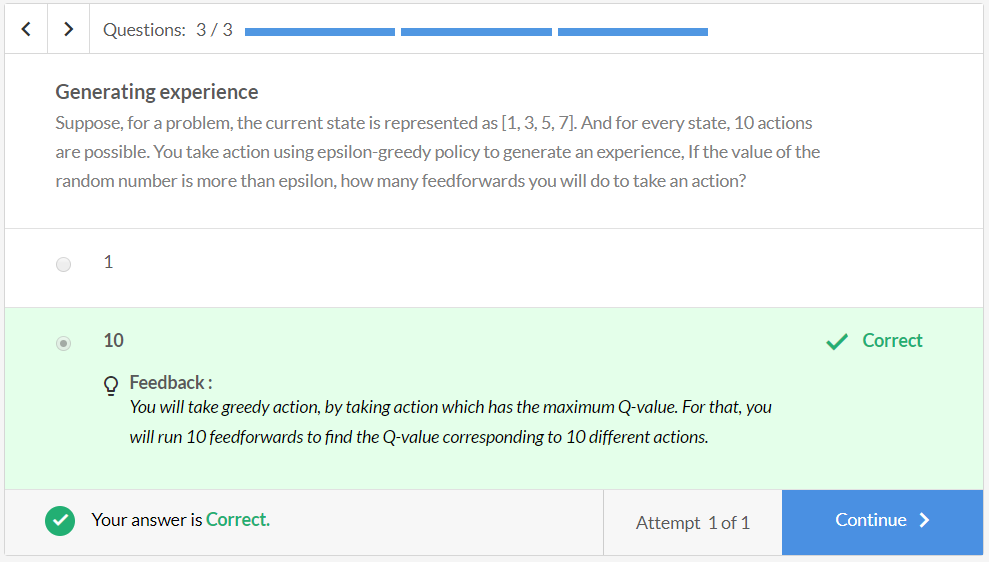
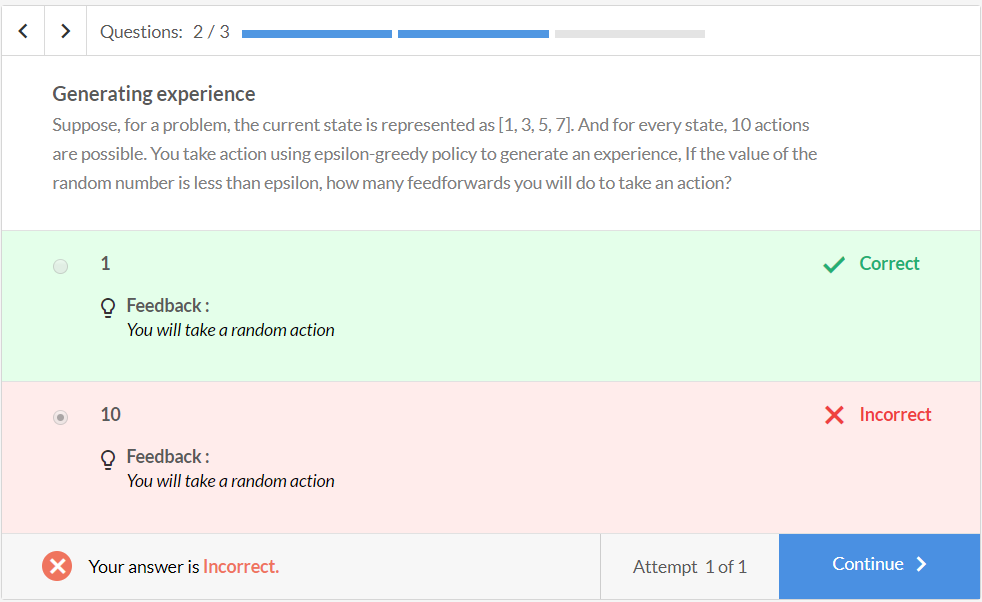
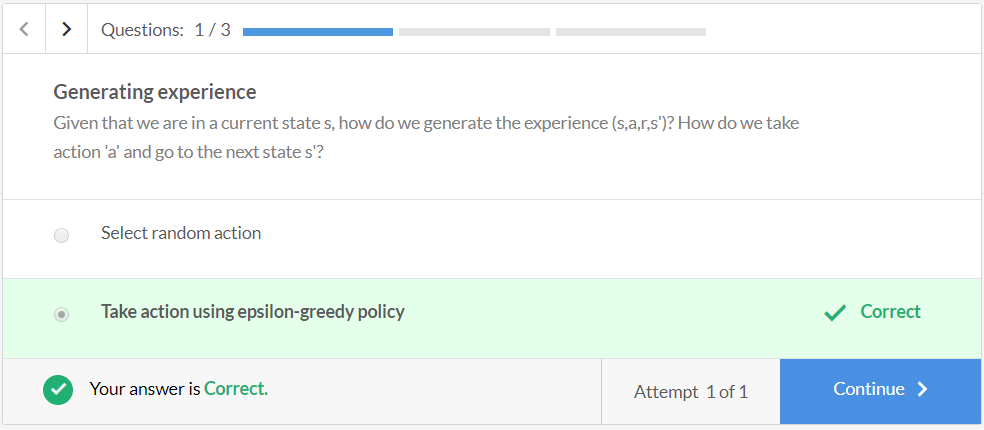
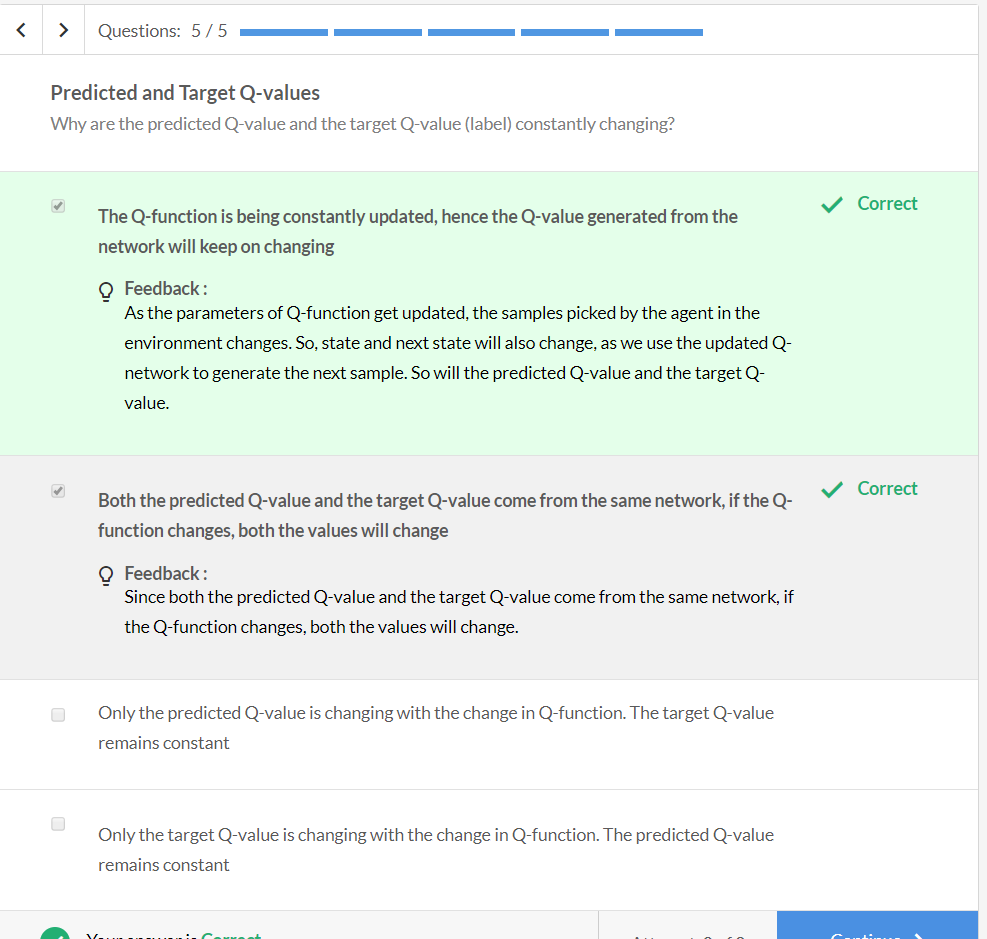
**Note: You are generating only one experience at every timestep while training on (say batch size = 100) 100 experiences at every timestep. This is so because it has been assumed that you must have generated 100 experiences previously and stored in memory. Training will start only after you have your 'first' 100 experiences.**

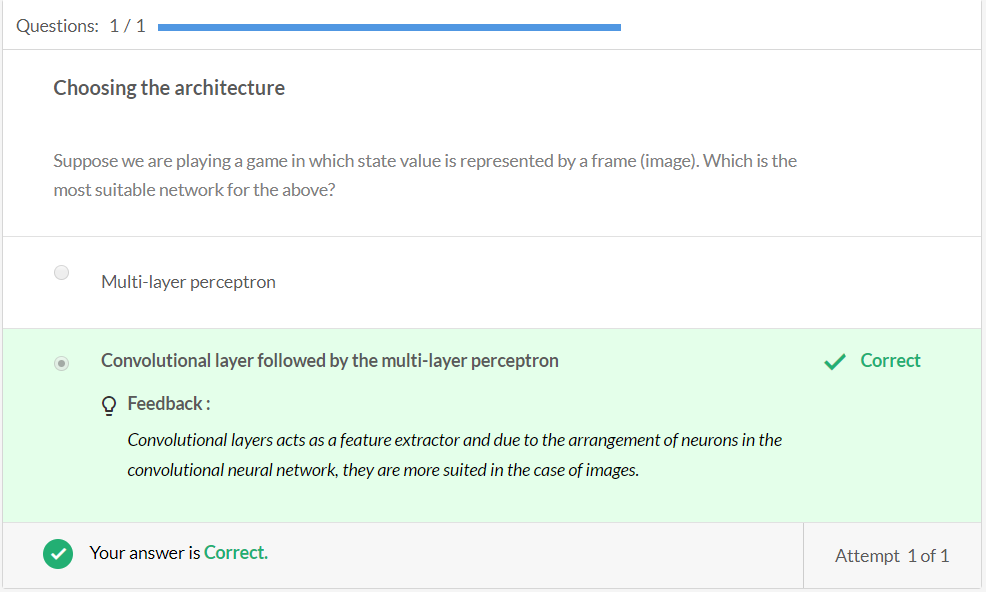
The Q-function is being updated at the end of every episode. Thus, you choose better actions after each update to generate the experiences. This will result in more optimal actions and next states, which will further improve the learning process (i.e. the policy).

 Both the input and label are changing with time which leads to a dynamic dataset.









# Summary

In this session, you learnt that it may not be possible to visit every state in case the possible state space is huge. If you have missed certain states while training and don't have their Q-values stored in a Q-table, how would you take an action if you encounter these states?

So, you use a function approximator to approximate the Q-value.  The neural network as function approximator is used because of its many advantages. The most important property of function approximator is the ability to 'generalise'. It should be able to find the optimal Q-value for an unvisited state, and obviously for a visited state.

Then you learnt that the nature of the data in reinforcement learning is non-i.i.d. So, to train the network, you generate an experience <s,a,s',r>, store it in memory and take a random sample of batch size to train the Q-value neural network.

The training process is, you randomly initialise a neural network, generate an experience and store it in the memory. Then you take random samples of batch size from the memory to find the Q(s,a) and Q(s',a) and train the network. The loss function is 'squared error'. After training for a single step in an episode, the parameters of the neural network will improve. So, use the updated neural network to generate the next experience.

