Deep SARSA

The estimates of the Q values are not stored in a table, but in the weights of a neural network. The state vector $s = [s_1, s_2]$ is the input to the neural network which will output $\hat{q}$ values in a vector:

$$\hat{q}(s) = [\hat{q}(s, a_1), \hat{q}(s, a_2), \hat{q}(s, a_3)]. \tag{1}$$

We take the Mean Squared of Temporal-Difference Errors as the cost function to minimize:

$$\hat{L}(w) = \frac{1}{|K|} \sum_{i=1}^{|K|} [R_i + \gamma \hat{q}(S_i^{'}, A_i^{'} \,|\, \theta_{target}) - \hat{q}(S_i, A_i \,|\, \theta)]^2. \tag{2}$$

where the "bootstrapped" portion $R_i + \gamma \hat{q}(S_i^{'}, A_i^{'} \,|\, \theta_{target})$ is the target we want to push our weights towards, and $\hat{q}(S_i, A_i \,|\, \theta)$ is the old state value.

Then we calculate the gradient of the Cost function w.r.t the parameters $\theta = [w_1, w_2, \cdots, w_n]$:

$$\nabla \hat{L}(\theta) = [\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \cdots, \frac{\partial L}{\partial w_n}] \tag{3}$$

which we will use to update the weights θ in a SGD step:

$$\theta \leftarrow -\alpha \nabla \hat{L}(\theta). \tag{4}$$

Replay Memory

During the episode, the agent stores the experiences (transitions) in a buffer/list:

$$B = \begin{bmatrix} (S_1, A_1, R_2, S_2) \\ (S_2, A_2, R_3, S_3) \\ \cdots \\ (S_{N-1}, A_{N-1}, R_N, S_N) \end{bmatrix} \tag{5}$$

which has a limited size and when it fills up, it replaces old transitions with new ones.

For SGD, we sample a random batch of transitions from the memory:

$$K = (S, A, R, S^{'}) \sim B. \tag{6}$$

Target Network

We make a copy of the neural network to calculate the targets/ choose the actions for bootstrapping the state-action value, so that the target network does not change with SGD (and remains stable) and consistent with chosen actions throughout the SGD:

$$\theta_{target} \leftarrow \theta. \tag{7}$$

You sync the target network with the optimal network every k episode.

> Even though the target network is different from actual network, it is still on policy, because you are using the same policy function to choose action from the values.