CartPole-v1 - Reinforcement learning

Roman Zvoda

Approach:

The neural network in my model is defined with three layers, and the Q-values are computed using `q_network`. The `loss_fn` calculates the difference between predicted and target Q-values for training. In this function we can also see the bellman equation in use.
When it comes to picking policy I decided to go with the `epsilon_greedy_policy` as it seemed to be the most optimal option which focuses on exploration and exploitation. The `update` function optimizes model parameters using computed gradients.

When it comes to configuration values such as `learning rate, gamma, batch_size, epsilon,epsilon_decay, epsilon_min, update_target_every` I was tweaking the values to get the “most optimal” result. Mostly playing with `epsilon and learning_rate` variables.

The training loop runs for 1500 episodes, collecting experiences and updating the network. The target network is updated periodically. Rewards are tracked and printed out for visual orientation on how model is performing. I print the average score of the model so its easier to compare 2 different models at the end.

Result:

Current model took around `2 hours` to train where initial 40% of the episodes were looped through pretty fast. As the model improved and started to rank at higher rewards each episode took longer to finish. Once training is done I pickle the data. To test the functionality I use the pickled data and run it in the environment without the training loop. I have managed to train my model to score `500` all the time I would say.