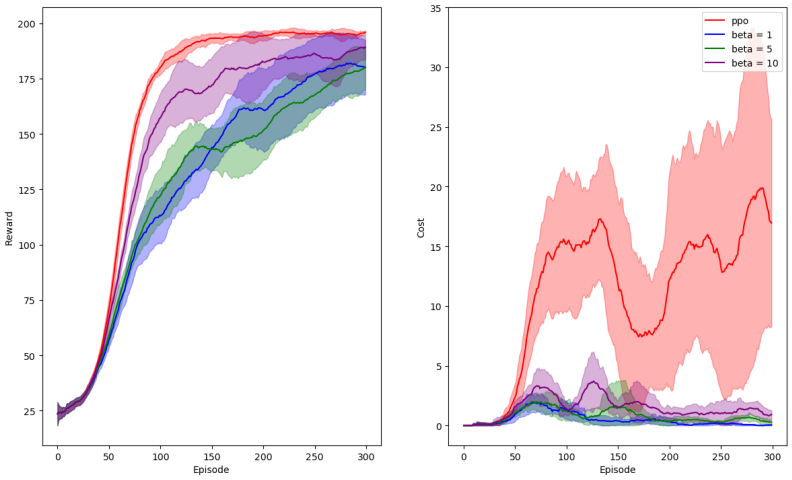
CS885 Assignment 3

Angelo Rajendram

November 2022

1 Part 1 Write Up - PPO Penalty

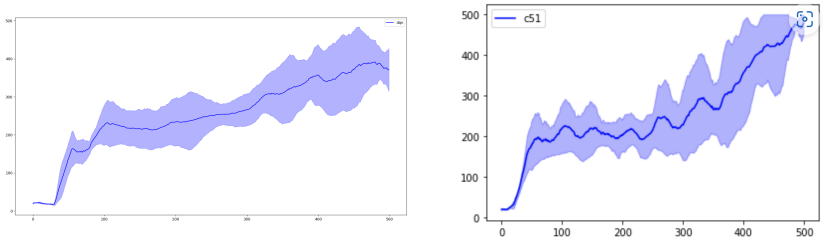


The plot above shows the results of PPO and PPO penalty for beta values of 0, 1, and 10. For the PPO penalty algorithm, a cost is invoked if the agent spends any time steps in the region $x \geq 1$. In the plot, we see that with regard to the rewards the PPO agent appears to perform the best, but the PPO penalty agents still approach a policy where the cart keeps the pole balanced for the entire duration of the episodes. This slower convergence to an optimal policy is due to the challenge of simultaneously having to satisfy the constraint. Looking at the plot of the costs incurred we see that plain PPO incurs high costs as it ignores the constraint, however, all PPO penalty agents keep the incurred costs close to zero while simultaneously collecting rewards.

Among the PPO penalty agents, we see a different rate of convergence depending on the value of beta. Higher values of beta correspond to a relaxation of the constraint during training. More specifically, the agent can spend more

time steps in the prohibited region before it is penalized during training if the value of beta is higher. This relaxation of the constraint permits the agent to spend longer stretches improving its ability to collect rewards before it has to refine its ability to satisfy the constraint and so we observe a fast convergence to an optimal policy with respect to the rewards. On the other hand, the costs incurred tend to be higher for higher beta values (though they are all still close to zero). This effect isn't particularly strong between beta values of 1 and 5, but is more clearly apparent when compared with a beta value of 10.

2 Part 2 Write Up - C51



(a) DQN (b) C51

The noisy cartpole environment is stochastic rather than deterministic. As we see in the plot above, the performance of C51 is better than DQN in this setting. For the noisy cart pole environment both DQN and C51 agents become better at balancing the pole, however, we see that DQN frequently fails to keep the pole balanced for 500 timesteps even after training. C51 successfully learns to consistently keep the pole balanced for the 500 timesteps upon training.

Learning distributions rather than the expected returns generally offer an advantage when approximations are being made. The C51 algorithm, unlike DQN, is able to mitigate instabilities or "chattering" in the training process by effectively averaging different distributions. The overestimation and underestimation of returns for state-action pairs that the standard DQN algorithm is susceptible to are mitigated to some extent with the distributional approach followed by the C51 algorithm. This is because overestimation and underestimation of particular return values and their corresponding probabilities tend to cancel out when computing average returns associated with an action. This explains the improved performance of C51 over DQN.