Policy Gradient RL Agent Plan

Heres my simplified plan for creating a policy gradient RL agent:

Implement a Policy Network

For the first part of the implementation, a simple neural network needs to be built to represent the policy. This network will take the current state of the environment as input and output a set of probabilities for each possible action. Using a softmax layer at the output ensures the probabilities sum to 1, allowing the agent to make decisions based on these probabilities.

Action Selection Using the Policy Network

Once the policy network is set up, it can be used to select actions. The network will output action probabilities, and an action is sampled based on these probabilities. This process introduces some randomness, allowing the agent to explore different actions rather than always selecting the same one.

Run Episodes and Collect Trajectories

Next, the agent is run through episodes, where it interacts with the environment by following the policy and taking actions. For each episode I need to record the agent's states, actions, and the rewards received. This data will be used later to train and improve the policy.

Compute Discounted Returns

After running an episode, the discounted returns for each step are calculated. This involves summing up future rewards, but applying a discount factor to give more importance to immediate rewards. This will overall make the agent balance short term goals with long term goals, but will prioritize short term.

Compute the Policy Gradient

Using the collected data from the episode, the next step is to compute the policy gradient. This involves taking the log probabilities of the actions the agent took, and weighting them by the corresponding returns.

Update the Policy

Once the policy gradient is calculated, the policy network’s weights are updated accordingly.

Build the Training Loop

Combine all previous steps into a training loop, where the agent runs through multiple episodes, collects data, calculates returns, computes gradients, and updates the policy. Over time, the agent learns to improve its policy and maximize its rewards in the environment.

Evaluate the Agent

Finally, after the agent has been trained, its performance is evaluated by running it without exploration to see how well the learned policy performs. The goal is to determine whether the agent can consistently maximize rewards after learning from multiple episodes.