Learning Stochastic Policies for Reinforcement Learning

Subhajit Roy

July 30, 2020

Reinforcement learning policies can either be deterministic or stochastic.

A deterministic policy is a function $Policy : State \mapsto Action$ from state to actions while a stochastic policy defines a distribution from states to actions $Policy : State \mapsto Prob(Action)$.

In many situations, a stochastic policy performs better than a deterministic policy, for instance:

1. When the environment is non-deterministic or stochastic, eg. ghosts in PacMan move randomly;

2. When the agent only has access to a partial state of the system. eg. in Poker the agent cannot see her opponent's cards.

Generally deep RL learns deterministic policies as a deep neural network (DNN). The question remains if we can learn stochastic policies efficiently. Let us perform the following experiments to understand the situation better.

Experiment #1: Learning DRL

Learn a deep reinforcement learning agent for any environment in the openai Gym (`https://gym.openai.com/`). Plot its performance with respect to the number of training rounds.

Experiment #2: Creating a case for stochastic policies

Hide a part of the state vector from the agent while training the DNN (i.e. use only a part of the state vector as features for learning the DNN). Plot its performance with respect to the number of training rounds.

Experiment #3: A naive stochastic policy

Change the policy such that the agent uses the DNN for the respective action with a 0.8 probability and, with 0.2 probability, selects one of the remaining actions uniformly at random. Plot its performance with respect to the number of training rounds.

Experiment #4: Learning the distribution of distortion

Construct a policy such that the agent uses the DNN for the respective action with probability p and, with probability $(1 - p)$, selects one of the remaining actions uniformly at random. Learn a neural network to predict this probability p for each state. Every time the agent gets a positive reward, the respective probability is bumped up, else it is pushed down. We may use some other learning algorithm also. Plot its performance with respect to the number of training rounds.