CS7011 : Topics in Reinforcement Learning

Summary of Latent Space Policies in HRL : Haarnoja, Hartikainen, Abbeel, Levine

Sahana Ramnath : EE15B109

1. The main idea

This paper (Haarnoja et al., 2018) proposes a new way to learn hierarchical deep neural network policies for reinforcement learning. Conventional hierarchical algorithms attempt to ensure that lower layers of the hierarchy are explicitly useful to the higher layers. In contrast to this, **SAC-LSP**(Soft Actor Critic Latent Space Policies) is a framework where each layer directly attempts to solve the task by optimizing a maximum entropy objective; if it's fully/partially successful, it makes the job easier for the layer above it.Each layer of the hierarchy corresponds to a policy with internal latent variables(sampled from a prior distribution while training that layer) which determine how the policy maps states to actions. Due to the maximum entropy objective, these latent variables are incorporated into the layer's policy, and the higher level layer can directly control the behavior of the lower layer through this latent space. Each layer is unconstrained in its relation to the environment and receives the full state as observation; further, by constraining the mapping from latent variables to actions to be invertible, the layers retain full expressivity; they are not restricted to use each other and the higher layers can undo any transformation of the action space imposed by the layers below. Each layer can be trained either on the true reward for the task, or on a lower level shaping reward. For example, while learning a complex navigation task, lower layers may receive a reward that promotes locomotion irrespective of direction, whereas higher layers may aim to reach a particular location.

2. Using Probabilistic Graphical Models

2.1. PGM for Control

This model is composed of factors for the dynamics $p(s_{t+1}|s_t, a_t)$ and for an action prior $p(a_t)$. The goal is to find an optimal trajectory given the reward function. So they attach to each state and action a binary RV O_t(*optimality variable*) denoting whether the time step was optimal. To solve the optimal control problem, they infer the posterior action distribution as $\pi^*(a_t|s_t) = p(a_t|s_t, O_{t:T} = true)$, which states that an optimal action is such that the optimality variable is active for the current state and for all of the future states. One way to incorporate the reward function into this is to choose $p(O_t|s_t, a_t) = exp(r(s_t, a_t))$. The distribution over optimal trajectories can be written as : $p(\tau|O_{0:T}) = p(s_0)\Pi_{t=0}^{T} p(a_t)p(s_{t+1}|s_t, a_t)exp(r(s_t, a_t))$.

2.2. RL via variational inference

The optimal action distribution obtained above cannot be directly used for 2 reasons : it would lead to an overly optimistic policy that assumes the dynamics can be modified to suit the agent; and would be intractable for continuous domains. Both issues are corrected using structural variational inference where the posterior is approximated using a probabilistic model that constrains the dynamics(to be equal to the actual one that can be sampled) and the policy.

2.3. Learning Latent Space Policies

The hierarchy is constructed by defining a stochastic base policy as a latent variable model.This has 2 factors : a conditional action distribution $\pi(a_t|s_t, h_t)$ (h_t is a latent random variable) and a prior $p(h_t)$. Addition of h_t leads to a new graphical model which can be conditioned on new optimality variables;the base policy is integrated into the MDP's transition structure which now gives a new,higher-level set of actions h_t. If the base policy succeeds in solving the task partially/completely, learning a related task with the action h_t is much easier. In this policy-augmented model, the transition model and action conditional together serve as a new(and likely easier) dynamical system. This can be repeated multiple times, learning a higher-level policy on top of the previous policy's latent space, thus leading to a *deep, hierarchical* policy. These policy layers are called *sub-policies*.

2.4. Representation of sub-policies

The sub-policies should be characterized by 3 properties : should be tractable, should be expressive(to not suppress information flow from higher to lower levels) and the conditional factor should be deterministic(since additional noise can degrade the performance).

3. The Algorithm

The algorithm begins by using the low-level actions in the environment according to the unknown dynamics, with $h_t^{(0)} = a_t$ as the lowest layer's actions. It trains each layer in turn, freezes its weights and trains new layers with the lower layer's latent variables as its action space. Each layer may be trained on the same maximum entropy objective and each layer simplifies the task for the layer above. If required, weak prior information/reward shaping can be incorporated, but the last layer's reward function must be for the actual goal task.

4. Experiments

Experiments were conducted on various continuous control benchmarks from the OpenAI suite. For all experiments, both single policies and hierarchical policies(composed of 2 sub policies) were used. The single policy model itself performed on par or better than all prior methods, and the two-level policy model outperformed the single level one by a large margin.The performance boost is more prominent if we train the base policy longer. Learning curves are provided in the paper for the Ant Maze and Humanoid tasks among others.

References

Haarnoja, Tuomas, Hartikainen, Kristian, Abbeel, Pieter, and Levine, Sergey. Latent space policies for hierarchical reinforcement learning. *arXiv preprint arXiv:1804.02808*, 2018.