Reninforcement Learning Project Report

Submitted By: Ankita Mehta

December 19 , 2017

1 INAC Algorithm over discrete and continuous actions

For this project, I implemented INAC (Incremental Natural Actor Critic Algorithm) (1), which is an advancement of the Natural Actor-Critic with Eligibility traces, over MDP's with continuous and discrete actions. The implementation has been written in python language. Actor-Critic methods implement generalised policy iteration where the actor aims at improving the current policy and the critic evaluates the current policy.

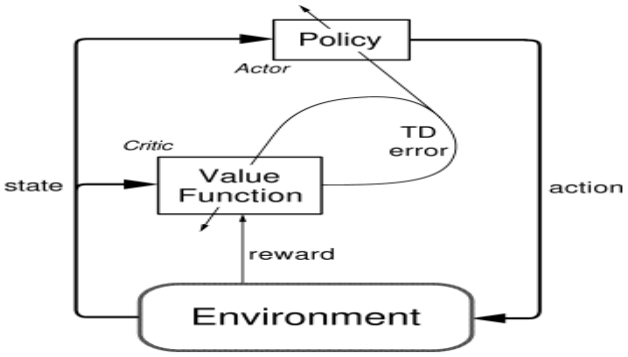

Figure 1: Actor Critic

Natural Actor-critic uses the policy-gradient framework to update its policy parameter (actor weights), θ. Both actor and critic updates their eligibility traces and weights at each time step. First, critic updates its weights using TD(λ) algorithm (2). Then the actor updates its weights (θ) based on the TD error δ and its eligibility traces. It uses the natural gradients considering distance metric as Euclidean.

INAC extends this algorithm by using the natural gradients with distance metric other than Euclidean distance, i.e Fisher Information Matrix. Natural gradient $\tilde{\nabla}_\theta J(\pi) = G(\theta)^{-1}\nabla_\theta J(\pi)$, where G(θ) is the Fisher Information Matrix.

$G(\theta) = \sum_{s\in S} d^\pi(s) \int_A \pi(a|s) \frac{\nabla_\theta \pi(a_t|s_t)}{\pi(a_t|s_t)} \frac{\nabla_\theta \pi(a_t|s_t)^\intercal}{\pi(a_t|s_t)}$.

INAC Update without average reward setting

$$
\begin{gathered}
Chose \quad action \quad a \quad according \quad to \quad \pi(a|s) \\
Take \quad action \quad a \quad in \quad s, \quad observe \quad s' \quad and \quad r \\
\delta \leftarrow r + \gamma v^\intercal \phi(s') - v^\intercal \phi(s) \\
e_v \leftarrow \gamma\lambda e_v + \phi(s) \\
v \leftarrow v + \alpha_{\text{critic}} \delta e_v \\
e_\theta \leftarrow \gamma\lambda e_\theta + \frac{\nabla_\theta \pi(a|s)}{\pi(s|s)} \\
w \leftarrow w - \alpha_{\text{critic}} \frac{\nabla_\theta \pi(a|s)}{\pi(s|s)} \frac{\nabla_\theta \pi(a|s)^\intercal}{\pi(a|s)} w + \alpha_{\text{critic}} \delta e_\theta \\
\theta \leftarrow \theta + \alpha_{\text{actor}} w
\end{gathered}
$$

Here, **v** is the weight vector of critic, θ is the weight vector of actor , e_v , e_θ are the eligibility traces of critic, actor respectively, $\phi(s)$ is a feature vector corresponding to the state s, $\frac{\nabla_\theta \pi(a|s)}{\pi(s|s)}$is the policy gradient.

NOTE: INAC update is done at every time step. At the start of every episode, all the eligibility traces are initialised to 0. When the agent reaches the end of episode, $\phi(s') = 0$ and then all weights are updated. The algorithm defined is independent of the structure of the policy distribution used in the policy.

As in the previous NAC algorithm, for INAC also the critic weights are updated with TD(λ). But for the actor, the vector w is updated and used as an estimate of the natural gradient to update the actor weights.

1.1 Discrete Actions

For discrete actions, policy taken is the softmax policy and then $\frac{\nabla \pi(a|\theta,s)}{\pi(s|s)}$ is calculated the taking the derivative of softmax policy.

$\pi(a|s,\theta) = \frac{\exp \phi(s,a)^\intercal.\theta}{\sum_{a'} \exp \phi(s,a')^\intercal.\theta}$

For a' = a (a is the action taken) $\frac{\nabla \pi(a|s,\theta)}{\pi(s|a,\theta)} = (1 - \pi(a|s,\theta)).\phi(s,a)$

For all other actions, $\frac{\nabla \pi(a|s,\theta)}{\pi(s|a,\theta)} = \sum_{a' \in A} \pi(a|s,\theta).\phi(s,a)$

For discrete actions, I implemented INAC on mountain car with discrete actions and Gridworld domain

1.2 Continuous Actions

For the continuous actions, policy is defined such that the actions are taken according to the Normal distribution with a probability density function.$\mathcal{N}(s,a) = \frac{1}{\sqrt{2\pi\sigma^2(s)}} \exp - \frac{(a-\mu(s))^2}{2\sigma^2(s)}$, where $\mu(s), \sigma(s)$ are the mean and standard deviation of the distribution $\pi(.|s)$

Here, $\mu(s) = \theta_\mu^\intercal \phi_\mu(s)$ and $\sigma(s) = \exp(\theta_\sigma^\intercal \phi_\sigma(s))$ and $\theta = (\theta_\mu^\intercal, \theta_\sigma^\intercal)^\intercal$.

Also, since the policy structure is normal distribution , so the compatible features $\frac{\nabla_\theta \pi(a|s)}{\pi(s|s)}$ is calculated the taking the derivative of normal distribution w.r.t μ, σ as defined in the paper (1)

Note: There was some instability issues coming since, the variance of the distribution was getting very-2 small- close to 0 and gradient was approaching nan, so I scaled the gradient w.r.t to the variance of the distribution i.e multiplying the gradient by $\alpha_{actor} * \sigma^2$ (3)

For continuous actions, I implemented INAC on mountain car with continuous actions and Pendulum domain.

2 Hyperparameter Tuning

For hyperparameter tuning, I firstly chose random values of $\alpha_{actor}, \alpha_{critic}, \lambda, fourier\_order$. I took $\gamma = 1$ and number_of_episodes 100 for all the experiments. Then I just tweaked one parameter (increase and decrease) and observe the plot of average rewards versus number_of_episodes. What was happening is, for very large and very small values of $\alpha_{actor}, \alpha_{critic}, fourier\_order$, the policy gradients were coming out to be nan. So I found a range of $\alpha_{actor}, \alpha_{critic}$ and the value of fourier_order where the agent was behaving properly. Then I started playing with the $\alpha_{actor}, \alpha_{critic}, \lambda$ by giving very small step sizes and fixing the fourier_order. Then by running a lot of experiments, I finally found the hyperparameters where all 4 agents were working fine. The hyperparameter values and their plots have been shown below.

3 Results

Below are the plots of number_of_episodes vs average rewards. Experiments are run for 100 episodes. **Note** Out of all the domains, hyperparameter tuning for pendulum domain took a lot of time. In this domain, for some trials 1/200, instability issue was coming. So those trials were just skipped.

3.1 Discrete Actions

1. Parameters for Mountain Car with discrete number of actions (3). $\alpha_{actor} = 0.001, \alpha_{critic} = 0.001, \lambda = 0.65$, $fourier\_order = 4$. Average of rewards is taken over 20 trials.

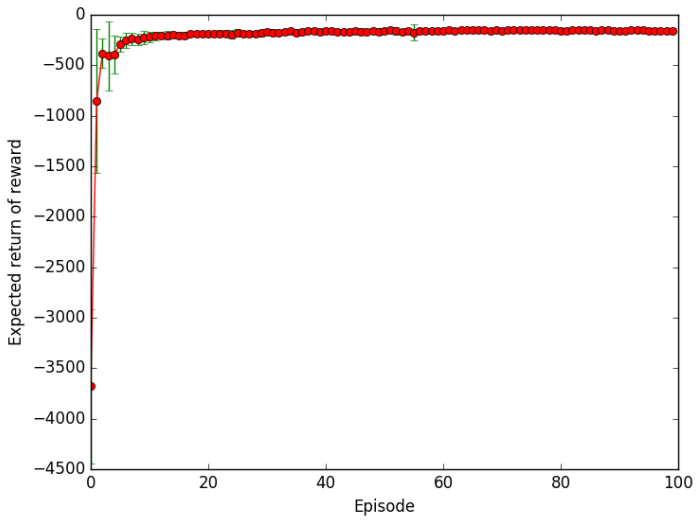

Figure 2: Mountain Car with discrete actions

How to run this code: Go to the directory : Project_report/Discrete_RL/
python INAC_parallel_wrapper.py -n 20 -l 0.65 -aa 0.001 -ac 0.001 -fo 4 -d mc -g 1

2. Parameters for Gridworld discrete number of actions (4). $\alpha_{actor} = 0.01, \alpha_{critic} = 0.01, \lambda = 0.9$, $fourier\_order = 0$. Average of rewards is taken over 50 trials.

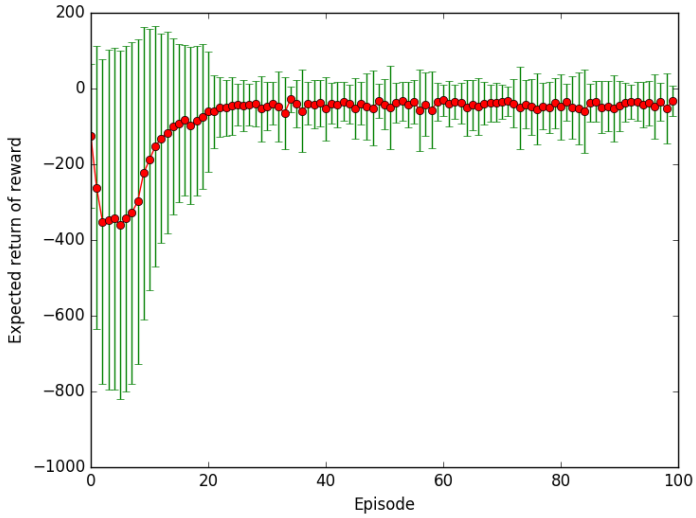

Figure 3: Gridworld with discrete actions

How to run this code: Go to the directory : Project_report/Discrete_RL/
python INAC_parallel_wrapper.py -n 50 -l 0.9 -aa 0.01 -ac 0.01 -fo 0 -d gw -g 1

3.2 Continuous Actions

1. Parameters for Mountain Car with continuous actions, [-1,1]. $\alpha_{actor} = 0.0001, \alpha_{critic} = 0.014, \lambda = 0.8$, $fourier\_order = 4$. Average of rewards is taken over 20 trials.

   How to run this code: Go to the directory : Project_report/Continuous_RL/
   python INAC_normal_wrapper.py -n 20 -l 0.8 -aa 0.0001 -ac 0.014 -fo 4 -d mc -g 1

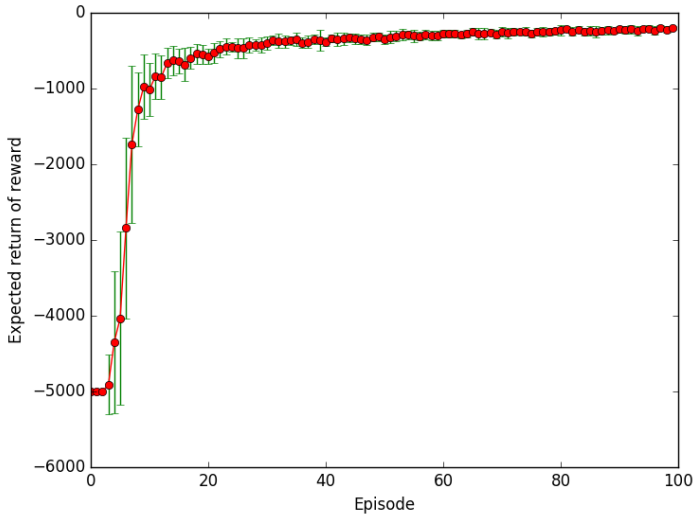

Figure 4: Mountain Car with continuous actions

2. Parameters for Pendulum with continuous actions (4). $\alpha_{actor} = 0.001, \alpha_{critic} = 0.001, \lambda = 0.7, fourier\_order = 4$. Average of rewards is taken over 1200 trials.

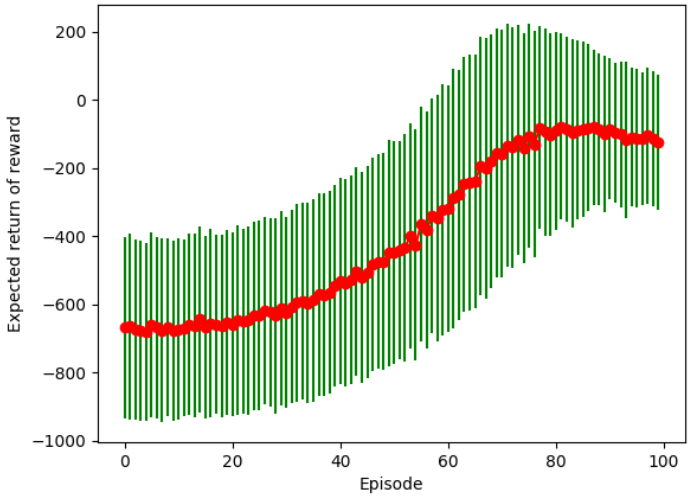

Figure 5: Pendulum with continuous actions

How to run this code: Go to the directory : Project_report/Continuous_RL/
python INAC_normal_wrapper.py -n 1200 -l 0.7 -aa 0.001 -ac 0.001 -fo 4 -d pen -g 1

References

[1] Thomas Degris, Patrick Pilarski, Richard Sutton. *Model-Free Reinforcement Learning with Continuous Action in Practice. American Control Conference, Jun 2012, Montreal, Canada. 2012. hal-0076428*

[2] Sutton, R. S. 1988 *Learning to Predict by the Methods of Temporal Differences.*

[3] Williams, R. 1992. *Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning.*