Reinforce Learning in Asset Allocation

Author: ZHONG, Ziyu & ZHONG, Jingwen   Student Number: 20923387 & 20930225

1 The Asset Allocation Problem

Consider the discrete-time asset allocation example in section 8.4 of *Rao and Jelvis*. Suppose the single-time-step return of the risky asset from time t to $t+1$ as $Y_t = a, prob = p$, and $b, prob = (1-p)$. Suppose that $T = 10$, use the TD method to find the Q function, and hence the optimal strategy.

1.1 Problem Setting

We are given wealth W_0 at time 0. At each of discrete time steps labeled $t = 0, 1, \ldots, T-1$, we are allowed to allocate the wealth W_t at time t to a portfolio of a risky asset and a riskless asset in an unconstrained manner with no transaction costs. The risky asset yields a random return Y_t over each single time step. The riskless asset yields a constant return denoted by r over each single time step (for a given $r \in \mathbb{R}$). We assume that there is no consumption of wealth at any time $t < T$, and that we liquidate and consume the wealth W_T at time T. So our goal is simply to maximize the Expected Utility of Wealth at the final time step $t = T$ by dynamically allocating $x_t \in \mathbb{R}$ in the risky asset and the remaining $W_t - x_t$ in the riskless asset for each $t = 0, 1, \ldots, T-1$. Assume the single-time-step discount factor is γ and that the Utility of Wealth at the final time step $t = T$ is given by the following CARA function:

$$U(W_T) = \frac{1 - e^{-cW_T}}{c} \text{for some fixed c} \neq 0$$

Thus, the problem is to maximize, for each $t = 0, 1, \ldots, T-1$, over choices of $x_t \in \mathbb{R}$, the value:

$$\mathbb{E}[\gamma^{T-t} \cdot \frac{1 - e^{-cW_T}}{c} | (t, W_t)]$$

Since γ^{T-t} and c are constants, this is equivalent to maximizing, for each $t = 0, 1, \ldots, T-1$, over choices of $x_t \in \mathbb{R}$, the value:

$$\mathbb{E}[-e^{-cW_T} | (t, W_t)]$$

We formulate this problem as a *Continuous States* and *Continuous Actions* discrete-time finite-horizon MDP by specifying its *State Transitions*, *Rewards* and *Discount Factor* precisely. The problem then is to solve the MDP's Control problem to find the Optimal Policy.

1.2 Analytic Solution of the Optimal Policy

A deterministic policy at time t (for all $t = 0, 1, \ldots T-1$) is denoted as π_t, and hence, we write: $\pi_t(W_t) = x_t$. Likewise, an optimal deterministic policy at time t (for all $t = 0, 1, \ldots, T-1$) is denoted as π_t^* , and hence, we write: $\pi_t^*(W_t) = x_t^*$.

Denote the random variable for the single-time-step return of the risky asset from time t to time $t+1$ as

$$Y_t = \begin{cases} a, \ prob = p \\ b, \ prob = (1-p) \end{cases}$$

for all $t = 0, 1, \ldots T-1$. So,

$$W_{t+1} = x_t \cdot (1 + Y_t) + (W_t - x_t) \cdot (1 + r) = x_t \cdot (Y_t - r) + W_t \cdot (1 + r) \tag{1}$$

for all $t = 0, 1, \ldots T - 1$.

We denote the Value Function at time t (for all $t = 0, 1, \ldots, T - 1$) for a given policy $\pi = (\pi_0, \pi_1, \ldots, \pi_{T-1})$ as:

$$V_t^\pi(W_t) = \mathbb{E}_\pi[-e^{-cW_T}|(t, W_t)] \tag{1}$$

We denote the Optimal Value Function at time t (for all $t = 0, 1, \ldots, T - 1$) as:

$$V_t^*(W_t) = \max_\pi V_t^\pi(W_t) = \max_\pi \{\mathbb{E}_\pi[-e^{-cW_T}|(t, W_t)]\} \tag{2}$$

The Bellman Optimality Equation is:

$$V_t^*(W_t) = \max_{x_t} Q_t^*(W_t, x_t) = \max_{x_t}\{\mathbb{E}_{Y_t}[V_{t+1}^*(W_{t+1})]\} \tag{3}$$

for all $t = 0, 1, \ldots T - 2$, and

$$V_{T-1}^*(W_{T-1}) = \max_{x_{T-1}} Q_{T-1}^*(W_{T-1}, x_{T-1}) = \max_{x_{T-1}}\{\mathbb{E}_{Y_{T-1}}[-e^{-cW_T}]\} \tag{4}$$

where Q_t^* is the Optimal Action-Value Function at time t for all t = $0, 1, \ldots, T - 1$.

We make an guess for the functional form of the Optimal Value Function as:

$$V_t^*(W_t) = -b_t \cdot e^{-c_t \cdot W_t} \tag{2}$$

where b_t, c_t are independent of the wealth W_t for all $t = 0, 1, \ldots, T - 1$. Next, we express the Bellman Optimality Equation using this functional form for the Optimal Value Function:

$$V_t^*(W_t) = \max_{x_t}\{\mathbb{E}_{Y_t}[-b_{t+1} \cdot e^{-c_{t+1} \cdot W_{t+1}}]\} \tag{5}$$

Using Equation (1), we can write this as:

$$V_t^*(W_t) = \max_{x_t}\{\mathbb{E}_{Y_t}[-b_{t+1} \cdot e^{-c_{t+1} \cdot (x_t \cdot (Y_t - r) + W_t \cdot (1+r))}]\} \tag{6}$$

The expectation of this exponential form (under the given distribution) evaluates to:

$$V_t^*(W_t) = \max_{x_t}\{-b_{t+1}(p \cdot e^{-c_{t+1} \cdot (x_t \cdot (a-r) + W_t \cdot (1+r))} + (1-p) \cdot e^{-c_{t+1} \cdot (x_t \cdot (b-r) + W_t \cdot (1+r))})\} \tag{3}$$

We can then infer the functional form for $Q_t^*(W_t, x_t)$ in terms of b_{t+1} and c_{t+1}:

$$Q_t^*(W_t, x_t) = -b_{t+1}[p \cdot e^{-c_{t+1} \cdot (x_t \cdot (a-r) + W_t \cdot (1+r))} + (1-p) \cdot e^{-c_{t+1} \cdot (x_t \cdot (b-r) + W_t \cdot (1+r))}] \tag{4}$$

Since the right-hand-side of the Bellman Optimality Equation (3) involves a max over x_t, we can say that the partial derivative of the term inside the max with respect to x_t is 0. This enables us to write the Optimal Allocation x_t^* in terms of c_{t+1}, as follows:

$$x_t^* = \frac{1}{c_{t+1}(a-b)} \ln \frac{(a-r)p}{(r-b)(1-p)} \tag{5}$$

We need to not that, assume $a > b$, r must satisfies $r \in (a, b)$. Otherwise, the optimal solution does not exist, i.e. $Q_t^*(W_t, x_t) \to \infty$ as $x_t \to \infty$ if $r \notin (a, b)$.

Next, we substitute this maximizing x_t^* in the Bellman Optimality Equation (3):

$$V_t^*(W_t) = -b_{t+1}(p \cdot e^{-\frac{a-r}{a-b}\ln\frac{(a-r)p}{(r-b)(1-p)}} + (1-p) \cdot e^{-\frac{b-r}{a-b}\ln\frac{(a-r)p}{(r-b)(1-p)}}) \cdot e^{-c_{t+1}(1+r)W_t} = -b_{t+1}K \cdot e^{-c_{t+1}(1+r)W_t} \quad (7)$$

Here we denote $K = p \cdot e^{-\frac{a-r}{a-b}\ln\frac{(a-r)p}{(r-b)(1-p)}} + (1-p) \cdot e^{-\frac{b-r}{a-b}\ln\frac{(a-r)p}{(r-b)(1-p)}}$.

And since

$$V_t^*(W_t) = -b_t \cdot e^{-c_t \cdot W_t} \tag{8}$$

we can write the following recursive equations for b_t and c_t:

$$\begin{aligned} b_t &= b_{t+1} \cdot K \\ c_t &= c_{t+1} \cdot (1+r) \end{aligned} \tag{9}$$

We can calculate b_{T-1} and c_{T-1} from the knowledge of the MDP Reward $-e^{-cW_T}$ (Utility of Terminal Wealth) at time $t = T$, which will enable us to unroll the above recursions for b_t and c_t for all $t = 0, 1, \ldots, T-2$. At time $t = T-1$:

$$\begin{aligned} b_{T-1} &= K \\ c_{T-1} &= c \cdot (1+r) \end{aligned} \tag{10}$$

Thus,

$$\begin{aligned} b_t &= K^{T-t} \\ c_t &= c \cdot (1+r)^{T-t} \end{aligned} \tag{11}$$

Substituting the solution for c_{t+1} in Equation (5) gives us the solution for the Optimal Policy:

$$x_t^* = \frac{1}{c(a-b)(1+r)^{T-t-1}} \cdot \ln\frac{(a-r)p}{(r-b)(1-p)} \tag{12}$$

Substituting the solutions for b_t and c_t in Equation (2) gives us the solution for the Optimal Value Function:

$$V_t^*(W_t) = -K^{T-t} \cdot e^{-c(1+r)^{T-t} \cdot W_t} \tag{13}$$

for all $t = 0, 1, \ldots, T-1$.

Substituting the solutions for b_{t+1} and c_{t+1} in Equation (4) gives us the solution for the Optimal Action-Value Function:

$$Q_t^*(W_t, x_t) = -K^{T-t-1}[p \cdot e^{-c(a-r)(1+r)^{T-t-1}x_t} + (1-p) \cdot e^{-c(b-r)(1+r)^{T-t-1}x_t}] \cdot e^{-c(1+r)^{T-t}W_t} \tag{6}$$

for all $t = 0, 1, \ldots, T-1$.

1.3 Numerical Experiment of Different Policies

In this chapter, we take numerical experiments to compare the performences of different policies, in order to dipict the properties of the problem and verify the our theoretical result above.

We give a concrete series of parameters, that is
$T = 10,\ a = 0.18,\ b = 0.02,\ r = 0.10,\ p = \frac{2}{3},\ c = 1,\ W_0 = 1.$

Then we have:

$$x_t^* = \frac{1}{c(a-b)(1+r)^{T-t-1}} \cdot \ln\frac{(a-r)p}{(r-b)(1-p)} = \frac{\ln 2}{0.16 \cdot 1.1^{T-t-1}} \approx \frac{4.33217}{1.1^{T-t-1}},$$

$$K = \frac{2}{3} \cdot 2^{-\frac{1}{2}} + \frac{1}{3} \cdot 2^{\frac{1}{2}} \approx 0.942809,$$

$$V_t^*(W_t) = -K^{T-t} \cdot e^{-c(1+r)^{T-t} \cdot W_t} \approx -0.942809^{T-t} \cdot e^{-1.1^{T-t} W_t}.$$

We choose 4 policies $\pi^*, \pi_1, \pi_2, \pi_3$ to take experiments, which are:

$$\pi^*(t) = x_t^* = \frac{\ln 2}{0.16 \cdot 1.1^{T-t-1}}$$

$$\pi_1(t) = 1$$

$$\pi_2(t) = 5$$

$$\pi_3(t) \sim \mathcal{N}(3, 1)$$

We expect that π^* will get the highest value, which is approximately $V_0^*(W_0) \approx -0.04147528$.

In [1]:

```
import numpy as np
import torch
import torch.nn as nn
import matplotlib.pyplot as plt

seed = 1
torch.manual_seed(seed)
np.random.seed(seed)
torch.set_default_dtype(torch.float)
```

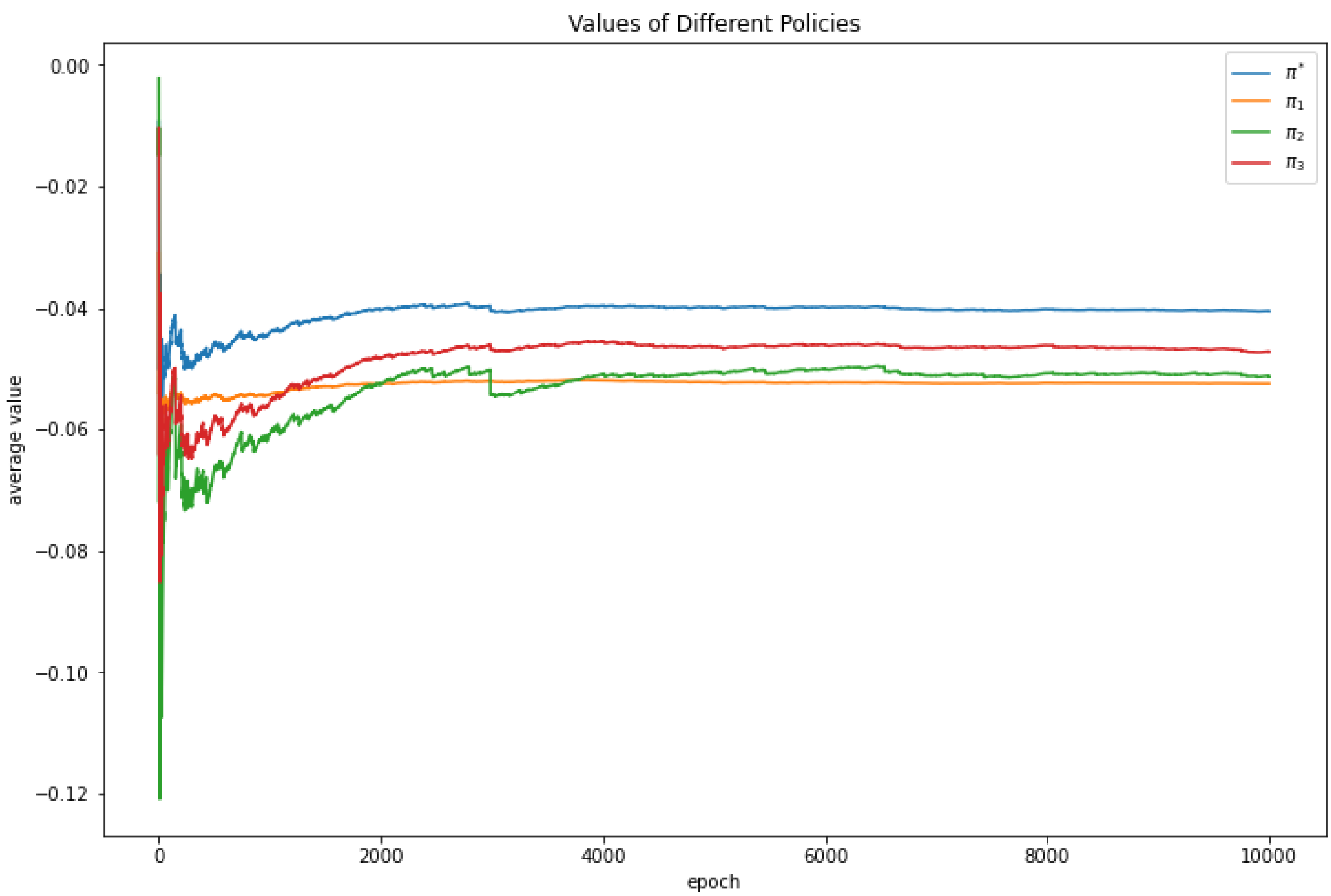
In [2]:

```
MAX_EPOCH = 10000
T = 10
a = 0.18
b = 0.02
r = 0.1
p = 2./3.
c = 1.
W_0 = 1.
pi_star_average_values = []
pi_1_average_values = []
pi_2_average_values = []
pi_3_average_values = []

for i in range(MAX_EPOCH):
    w0, w1, w2, w3 = 4 * [W_0]
    for t in range(T):
        y = np.random.choice([a, b], p=[p, 1-p])
        w0 = np.log(2)/(0.16 * np.power(1.1,T-t-1)) * (y - r) + w0 * (1 + r)
        w1 = 1. * (y - r) + w1 * (1 + r)
        w2 = 4. * (y - r) + w2 * (1 + r)
        w3 = (3 + np.random.randn()) * (y - r) + w3 * (1 + r)
    pi_star_value = -np.exp(-c * w0)
    pi_1_value = -np.exp(-c * w1)
    pi_2_value = -np.exp(-c * w2)
    pi_3_value = -np.exp(-c * w3)
    if i > 0:
        pi_star_value = i/(i+1) * pi_star_average_values[-1] + pi_star_value/(i+1)
        pi_1_value = i/(i+1) * pi_1_average_values[-1] + pi_1_value/(i+1)
        pi_2_value = i/(i+1) * pi_2_average_values[-1] + pi_2_value/(i+1)
        pi_3_value = i/(i+1) * pi_3_average_values[-1] + pi_3_value/(i+1)
```

```
    pi_star_average_values.append(pi_star_value)
    pi_1_average_values.append(pi_1_value)
    pi_2_average_values.append(pi_2_value)
    pi_3_average_values.append(pi_3_value)

plt.figure(figsize=(12, 8))
plt.plot(pi_star_average_values,label='$\pi^*$')
plt.plot(pi_1_average_values,label='$\pi_1$')
plt.plot(pi_2_average_values,label='$\pi_2$')
plt.plot(pi_3_average_values,label='$\pi_3$')
plt.xlabel('epoch')
plt.ylabel('average value')
plt.title('Values of Different Policies')
plt.legend()
plt.show()
```



In [3]:

```
#  The average value of the optimal policy after 10000 epochs
pi_star_average_values[-1]
```

Out[3]:

```
-0.040529294351449724
```

Conclusion:

From this figure, we can clearly see the policy π^* dominates the others. The average value of 10000 epochs $\approx$ -0.040529, which is closed to the theoretical value -0.041475.

Meanwile, we should pay attention that, the variance of the values are not small. And it will get significantly larger when $|x_t|$ is larger, since $T = 10$ could amplify the effect of investments.

2 Reinforce Learning Algorithm Designing

As the MDP problem has a continuous state space ($W_t \in \mathbb{R}$) and a continuous action space ($x_t \in \mathbb{R}$), we give up table methods which can not precisely model this problem. Besides, we already have the knowledge of the deterministic policy. It naturally comes out that **DDPG** *(Deep Deterministic Policy Gradient)* fits the problem well. In next chapters, we will introduce DDPG and show how to deploy it for our problem.

2.1 Summary of DDPG

Deep Deterministic Policy Gradient (DDPG) is an algorithm which concurrently learns a Q-function and a policy. It uses off-policy data and the Bellman equation to learn the Q-function, and uses the Q-function to learn the policy.

This approach is closely connected to Q-learning, and is motivated the same way: if you know the optimal action-value function $Q^*(s,a)$, then in any given state, the optimal action $a^*(s)$ can be found by solving

$$a^*(s) = \arg\max_a Q^*(s,a).$$

DDPG interleaves learning an approximator to $Q^*(s,a)$ with learning an approximator to $a^*(s)$, and it does so in a way which is specifically adapted for environments with continuous action spaces. But what does it mean that DDPG is adapted specifically for environments with continuous action spaces? It relates to how we compute the max over actions in $\max_a Q^*(s,a)$.

When there are a finite number of discrete actions, the max poses no problem, because we can just compute the Q-values for each action separately and directly compare them. (This also immediately gives us the action which maximizes the Q-value.) But when the action space is continuous, we can’t exhaustively evaluate the space, and solving the optimization problem is highly non-trivial. Using a normal optimization algorithm would make calculating $\max_a Q^*(s,a)$ a painfully expensive subroutine. And since it would need to be run every time the agent wants to take an action in the environment, this is unacceptable.

Because the action space is continuous, the function $Q^*(s,a)$ is presumed to be differentiable with respect to the action argument. This allows us to set up an efficient, gradient-based learning rule for a policy $\mu(s)$ which exploits that fact. Then, instead of running an expensive optimization subroutine each time we wish to compute $\max_a Q(s,a)$, we can approximate it with $\max_a Q(s,a) \approx Q(s,\mu(s))$.

DDPG use an *Actor-Critic* network to estimate Action-Value Function $Q^*(s,a)$ (by *Critic*), and actions $a = \mu(s)$ (by *Actor*). *Critic* is updated by minimizing mean-squared TD error, and *Actor* is updated by maxizing Action-Value Function $Q^*(s,\mu(s))$. Replay buffers and soft-replacement are also used in DDPG. The pseudocode is showed as follow:

Algorithm 1 Deep Deterministic Policy Gradient

Input: initial policy parameters θ, Q-function parameters ϕ, empty replay buffer $\mathcal{D}$
Set target parameters equal to main parameters $\theta_{\text{targ}} \leftarrow \theta$, $\phi_{\text{targ}} \leftarrow \phi$
repeat
  Observe state s and select action $a = \text{clip}(\mu_\theta(s) + \epsilon, a_{Low}, a_{High})$, where $\epsilon \sim \mathcal{N}$
  Execute a in the environment
  Observe next state s', reward r, and done signal d to indicate whether s' is terminal
  Store (s, a, r, s', d) in replay buffer $\mathcal{D}$
  If s' is terminal, reset environment state.
  if it's time to update **then**
    for however many updates **do**
      Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
      Compute targets

$$y(r, s', d) = r + \gamma(1 - d)Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))$$

      Update Q-function by one step of gradient descent using

$$\nabla_\phi \frac{1}{|B|} \sum_{(s,a,r,s',d)\in B} (Q_\phi(s, a) - y(r, s', d))^2$$

      Update policy by one step of gradient ascent using

$$\nabla_\theta \frac{1}{|B|} \sum_{s\in B} Q_\phi(s, \mu_\theta(s))$$

      Update target networks with

$$\phi_{\text{targ}} \leftarrow \rho\phi_{\text{targ}} + (1 - \rho)\phi$$
$$\theta_{\text{targ}} \leftarrow \rho\theta_{\text{targ}} + (1 - \rho)\theta$$

    end for
  end if
until convergence

2.2 Setup in Asset Allocation Problem

We inherit parameters setting in chapter **1.3**, that is

$$T = 10,\ a = 0.18,\ b = 0.02,\ r = 0.10,\ p = \frac{2}{3},\ c = 1,\ W_0 = 1.$$

To specify feature functions and structure of network, we need to leverage the functional form of the closed-form solution for the Action-Value function (6). We observe that we can write this as:

$$Q_t^*(W_t, x_t) = -e^{-A(T-t-1)-B(1+r)^{T-t}W_t}[C \cdot e^{-D(1+r)^{T-t-1}x_t} + (1 - C) \cdot e^{-E(1+r)^{T-t-1}x_t}]$$

where

$$A = -\ln K = -\ln(\frac{2}{3} \cdot 2^{-\frac{1}{2}} + \frac{1}{3} \cdot 2^{\frac{1}{2}}) \approx 0.0588915,$$

$$B = c = 1,$$

$$C = p = \frac{2}{3},$$

$D = c(a - r) = 0.08,$

$E = c(b - r) = -0.08.$

And the Optimal Policy is:

$$x_t^* = \frac{F}{(1+r)^{T-t-1}}$$

where

$F = \frac{1}{c(a-b)} \cdot \ln \frac{(a-r)p}{(r-b)(1-p)} = \frac{\ln 2}{0.16} \approx 4.33217.$

Then we can select following feature functions to approximate $Q_t^*(W_t, x_t)$:

$\phi_1((t, W_t)) = T - t - 1$

$\phi_2((t, W_t)) = (1+r)^{T-t} W_t$

$\phi_3((t, x_t)) = (1+r)^{T-t-1} x_t$

So $Q_t^*(W_t, x_t) = Q_t^*(\phi_1, \phi_2, \phi_3) = -e^{-A\phi_1 - B\phi_2}[C \cdot e^{-D\phi_3} + (1-C) \cdot e^{-E\phi_3}]$ is chosen to be *Critic*.

And if we restrict $x_t = \frac{F}{(1+r)^{T-t-1}}$, we can infer that $\phi_3((t, x_t)) \equiv F$. Thus we can choose $\mu_t = (1+r)^{T-t-1} x_t = F$ to be *Actor*,

then $Q_t^*(\phi_1, \phi_2, \phi_3) = Q_t^*(\phi_1, \phi_2, \mu_t)$, which is a stantard DDPG formation.

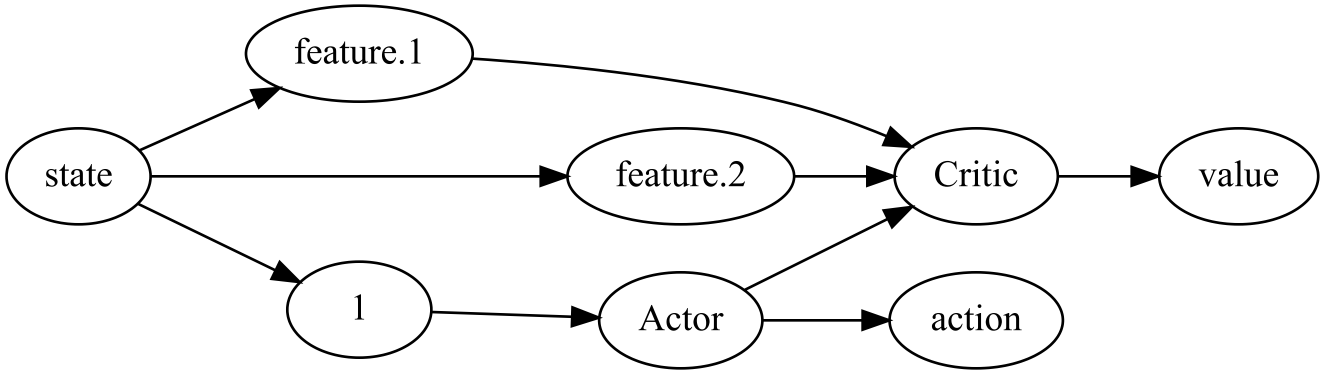
A forward computation graph is showed below:

In [3]:

```python
from graphviz import Digraph

g = Digraph('G')
g.graph_attr['rankdir'] = 'LR'
g.node('state', label='state')
g.node('feature1', label='feature.1')
g.node('feature2', label='feature.2')
g.node('feature3', label='1')
g.node('Critic', label='Critic')
g.node('Actor', label='Actor')
g.node('action', label='action')
g.node('value', label='value')
g.edge('state', 'feature1')
g.edge('state', 'feature2')
g.edge('state', 'feature3')
g.edge('feature1', 'Critic')
g.edge('feature2', 'Critic')
g.edge('feature3', 'Actor')
g.edge('Actor', 'Critic')
g.edge('Actor', 'action')
g.edge('Critic', 'value')
g
```

Out[3]:



2.3 Code Implememt

In [4]:

```
# import package and set random seed
import torch
import numpy as np
import torch.nn as nn

# set random state = 1
seed = 1
torch.manual_seed(seed)
np.random.seed(seed)
torch.set_default_dtype(torch.float)
```

In [5]:

```
# Define Actor
class Actor(nn.Module):
    def __init__(self, action_state_dim, action_dim):
        super(Actor, self).__init__()
        # set parameters as we said
        self.F = torch.nn.Parameter(torch.tensor(4., requires_grad=True))

    def replace(self, actor, tau):
        # function to implement soft and hard replacement
        para_iter = self.parameters()
        for para in actor.parameters():
            para_this = next(para_iter)
            para_this.data = (1.-tau) * para_this.data + tau * para.data

    def forward(self, s):
        return self.F * torch.ones_like(s)
```

In [6]:

```
# Define Critic
class Critic(nn.Module):

    def __init__(self, state_dim, action_dim):
        super(Critic, self).__init__()
        # set parameters as we said
        self.A = torch.nn.Parameter(torch.tensor(0.1, requires_grad=True))
        self.B = torch.nn.Parameter(torch.tensor(10.0, requires_grad=True))
        self.C = torch.nn.Parameter(torch.tensor(0.6, requires_grad=True))
        self.D = torch.nn.Parameter(torch.tensor(0.1, requires_grad=True))
        self.E = torch.nn.Parameter(torch.tensor(-0.1, requires_grad=True))

    def replace(self, crit, tau):
        # function to implement soft and hard replacement
        para_iter = self.parameters()
        for para in crit.parameters():
            para_this = next(para_iter)
            para_this.data = (1.-tau) * para_this.data + tau * para.data
```

```
    def forward(self, s, a):

        s1, s2 = torch.split(s, split_size_or_sections=1, dim=1)
        return -torch.exp(-self.A * s1 - self.B * s2) * (self.C * torch.exp(-self.D * a) +
```

In [7]:

```
class DDPG(object):
    def __init__(self, state_dim, action_state_dim, action_dim, replacement,memory_capacit
        super(DDPG, self).__init__()
        self.state_dim = state_dim
        self.action_state_dim = action_state_dim
        self.action_dim = action_dim
        self.memory_capacity = memory_capacity
        self.replacement = replacement
        self.t_replace_counter = 0
        self.gamma = gamma
        self.penal = np.log(2)/0.16
        self.lr_a = lr_a
        self.lr_c = lr_c
        self.batch_size = batch_size

        # Replay Buffer
        self.memory = np.zeros((memory_capacity, (state_dim + action_state_dim) * 2 + acti
        self.pointer = 0
        # Define Actor network
        self.actor = Actor(action_state_dim, action_dim)
        self.actor_target = Actor(action_state_dim, action_dim)
        # Define Critic network
        self.critic = Critic(state_dim,action_dim)
        self.critic_target = Critic(state_dim,action_dim)
        # Define Optimizer
        self.aopt = torch.optim.Adam(self.actor.parameters(), lr=lr_a)
        self.copt = torch.optim.Adam(self.critic.parameters(), lr=lr_c)
        # Define Loss Function
        self.mse_loss = nn.MSELoss()

    def sample(self):
        indices = np.random.choice(self.memory_capacity, size=self.batch_size)
        return self.memory[indices, :]

    def choose_action(self, s):
        s = torch.FloatTensor(s)
        action = self.actor(s)
        return action.detach().numpy()

    def learn(self):

        # soft replacement and hard replacement
        if self.replacement['name'] == 'soft':
            # soft replacement means take replacement every step
            tau = self.replacement['tau']
            self.actor_target.replace(self.actor, tau)
            self.critic_target.replace(self.critic, tau)

        else:
            # hard replacement means take replacement after a number of steps
            if self.t_replace_counter % self.replacement['rep_iter'] == 0:
                self.t_replace_counter = 0
                self.actor_target.replace(self.actor, 1.)
                self.critic_target.replace(self.critic, 1.)
            self.t_replace_counter += 1

        # Sample bacth data from replay buffer
        bm = self.sample()
```

```
        bs = torch.FloatTensor(bm[:, : self.state_dim])
        bas = torch.FloatTensor(bm[:, self.state_dim: self.state_dim + self.action_state_c
        ba = torch.FloatTensor(bm[:, self.state_dim + self.action_state_dim: self.state_di
        br = torch.FloatTensor(bm[:, -self.state_dim - self.action_state_dim - 2: -self.st
        bs_ = torch.FloatTensor(bm[:,-self.state_dim - self.action_state_dim - 1: -self.ac
        bas_ = torch.FloatTensor(bm[:,-self.action_state_dim-1: -1])
        bd = torch.FloatTensor(bm[:,-1:])

        # Train Actor
        a = self.actor(bas)
        q = self.critic(bs, a)
        a_loss = -torch.mean(q) + 0.00001 * torch.mean(torch.abs(a - self.penal))
        self.aopt.zero_grad()
        a_loss.backward(retain_graph=True)
        self.aopt.step()

        # Train Critic
        a_ = self.actor_target(bas_)
        q_ = self.critic_target(bs_, a_)
        q_target = br + self.gamma * (1 - bd) * q_
        q_eval = self.critic(bs, ba)
        td_error = self.mse_loss(q_target,q_eval)
        self.copt.zero_grad()
        td_error.backward()
        self.copt.step()

    def store_transition(self, s, sa, a, r, s_, sa_, d):
        transition = np.hstack((s, sa, a, [r], s_, sa_, [d]))
        index = self.pointer % self.memory_capacity
        self.memory[index, :] = transition
        self.pointer += 1
```

In [8]:

```
# Create the environment of asset dynamics
class Env:
    def __init__(self, T, a, b, r, p, c, W_0):
        self.t = 0
        self.W = W_0
        self.W_0 = W_0
        self.T = T
        self.a = a
        self.b = b
        self.r = r
        self.p = p
        self.c = c
        self.done = False

    def utility(self, W):
        return -np.exp(-self.c * W)

    def reset(self):
        self.t = 0
        self.W = self.W_0
        self.done = False
        return [self.t, self.W]

    def step(self, x):
        if self.done:
            return False
        W_new = x[0] * (np.random.choice((self.a, self.b), p=(self.p, 1-self.p)) - self.r)
        if self.t == 0:
            reward = self.utility(W_new)
        else:
            reward = self.utility(W_new) - self.utility(self.W)
        self.t += 1
```

```python
        self.W = W_new
        if self.t > self.T - 1:
            self.done = True
        return [self.t, self.W], reward, self.done
```

In [9]:

```python
# Define feature function
def feature_select(s, is_actor = False, T = 10, r = 0.1):
    if is_actor:
        return [1]
    else:
        return [T - s[0] - 1, np.power(1+r, T-s[0]) * s[1]]
```

In [10]:

```python
if __name__ == '__main__':

    # hyper parameters
    MAX_EPISODES = 15000
    MAX_EP_STEPS = 20
    MEMORY_CAPACITY = 5000
    BATCH_SIZE=2000
    LR_A=0.0001
    LR_C=0.0001
    REPLACEMENT = [
        dict(name='soft', tau=0.01),
        dict(name='hard', rep_iter=600)
    ][0]  # you can try different target replacement strategies

    T = 10
    a = 0.18
    b = 0.02
    r = 0.1
    p = 2./3.
    c = 10.
    W_0 = 1.

    # train
    env = Env(T, a, b, r, p, c, W_0)

    s_dim = 2
    as_dim = 1
    a_dim = 1
    ddpg = DDPG(state_dim=s_dim,
                action_state_dim = as_dim,
                action_dim=a_dim,
                replacement=REPLACEMENT,
                lr_a=LR_A,
                lr_c=LR_C,
                memory_capacity=MEMORY_CAPACITY,
                batch_size=BATCH_SIZE)

    average_values = []
    last_investments = []
    for i in range(MAX_EPISODES):
        s = env.reset()
        ep_reward = 0
        for j in range(MAX_EP_STEPS):
            # Add exploration noise
            s_s = feature_select(s, is_actor = False, T = env.T, r = env.r)
            a_s = feature_select(s, is_actor = True, T = env.T, r = env.r)
            a = ddpg.choose_action(a_s)
            x = a * np.power(1 + env.r, env.t - env.T + 1)

            s_, reward, done = env.step(x)
```

```python
            s_s_ = feature_select(s_, is_actor = False, T = env.T, r = env.r)
            a_s_ = feature_select(s_, is_actor = True, T = env.T, r = env.r)
            ddpg.store_transition(s_s, a_s, a, reward, s_s_, a_s_, done)

            ddpg.learn()

            s = s_
            ep_reward += reward
            if done or j == MAX_EP_STEPS - 1:
                if i > 0:
                    ep_reward = i/(i+1) * average_values[-1] + ep_reward/(i+1)
                average_values.append(ep_reward)
                last_investments.append(x)
                if i % 50 == 0:
                    print('Episode:', i, ' Total Reward:', ep_reward, 'Last investment:',
                break
```

```
Episode: 0  Total Reward: -5.438570596803339e-12 Last investment: [3.999101]
Episode: 50  Total Reward: -2.1846168584682578e-05 Last investment: [3.9576638]
Episode: 100  Total Reward: -1.1031263943249726e-05 Last investment: [3.9439487]
Episode: 150  Total Reward: -7.378588364359447e-06 Last investment: [3.9722977]
Episode: 200  Total Reward: -5.553496723216511e-06 Last investment: [4.03059]
Episode: 250  Total Reward: -4.456000836666053e-06 Last investment: [4.0934844]
Episode: 300  Total Reward: -3.7342421689402924e-06 Last investment: [4.1520276]
Episode: 350  Total Reward: -1.0229586855830028e-05 Last investment: [4.2232547]
Episode: 400  Total Reward: -8.965462548697166e-06 Last investment: [4.27858]
Episode: 450  Total Reward: -0.010939102337999215 Last investment: [4.3145504]
Episode: 500  Total Reward: -0.009847386513551055 Last investment: [4.325708]
Episode: 550  Total Reward: -0.008953794321022116 Last investment: [4.3257113]
Episode: 600  Total Reward: -0.008208895639781026 Last investment: [4.3257113]
Episode: 650  Total Reward: -0.0075871092112542214 Last investment: [4.3257113]
Episode: 700  Total Reward: -0.007045945960909343 Last investment: [4.3257113]
Episode: 750  Total Reward: -0.006576864040662489 Last investment: [4.325695]
Episode: 800  Total Reward: -0.006166337186280033 Last investment: [4.325512]
Episode: 850  Total Reward: -0.005804037723593518 Last investment: [4.325299]
Episode: 900  Total Reward: -0.005481961445857305 Last investment: [4.324767]
Episode: 950  Total Reward: -0.00519374057862547 Last investment: [4.324251]
Episode: 1000  Total Reward: -0.004934312988319233 Last investment: [4.324546]
Episode: 1050  Total Reward: -0.004704933904329016 Last investment: [4.325009]
Episode: 1100  Total Reward: -0.004496399727717704 Last investment: [4.323857]
Episode: 1150  Total Reward: -0.0043010835690591 Last investment: [4.3203835]
Episode: 1200  Total Reward: -0.004122020990697406 Last investment: [4.3166714]
Episode: 1250  Total Reward: -0.00395727195468826 Last investment: [4.3132405]
Episode: 1300  Total Reward: -0.007846680199919019 Last investment: [4.297789]
Episode: 1350  Total Reward: -0.0075562775317443015 Last investment: [4.2747736]
Episode: 1400  Total Reward: -0.00728660638963632 Last investment: [4.250627]
Episode: 1450  Total Reward: -0.007035517278555572 Last investment: [4.2219024]
Episode: 1500  Total Reward: -0.006801156276623188 Last investment: [4.189959]
Episode: 1550  Total Reward: -0.006581907735219533 Last investment: [4.1568356]
Episode: 1600  Total Reward: -0.006376351606439861 Last investment: [4.123664]
Episode: 1650  Total Reward: -0.006183245864283505 Last investment: [4.0965667]
Episode: 1700  Total Reward: -0.006001492614405832 Last investment: [4.0641665]
Episode: 1750  Total Reward: -0.0058301193265077305 Last investment: [4.0277586]
Episode: 1800  Total Reward: -0.005668261497867927 Last investment: [4.0173693]
Episode: 1850  Total Reward: -0.005515148008895902 Last investment: [4.02166]
Episode: 1900  Total Reward: -0.0053700900073775605 Last investment: [4.0272017]
Episode: 1950  Total Reward: -0.005232467094106552 Last investment: [4.034358]
Episode: 2000  Total Reward: -0.0051017207934072675 Last investment: [4.0435123]
Episode: 2050  Total Reward: -0.004978819045928282 Last investment: [4.044198]
Episode: 2100  Total Reward: -0.004860335239154219 Last investment: [4.021168]
Episode: 2150  Total Reward: -0.004747358653237211 Last investment: [4.0004435]
Episode: 2200  Total Reward: -0.004639513161967505 Last investment: [3.9791176]
Episode: 2250  Total Reward: -0.0045364602676905615 Last investment: [3.9619362]
Episode: 2300  Total Reward: -0.0044378844305669775 Last investment: [3.9440808]
Episode: 2350  Total Reward: -0.0043435028726934045 Last investment: [3.9208148]
```

```
Episode: 2400  Total Reward: -0.0042530509202021935 Last investment: [3.8991456]
Episode: 2450  Total Reward: -0.004166289381110785 Last investment: [3.8791938]
Episode: 2500  Total Reward: -0.004082997441276353 Last investment: [3.867558]
Episode: 2550  Total Reward: -0.0040029715504276584 Last investment: [3.8665884]
Episode: 2600  Total Reward: -0.003926020925739661 Last investment: [3.8891745]
Episode: 2650  Total Reward: -0.003851973554770422 Last investment: [3.916203]
Episode: 2700  Total Reward: -0.0037809910813566363 Last investment: [3.9467196]
Episode: 2750  Total Reward: -0.0037122714570709055 Last investment: [3.9812803]
Episode: 2800  Total Reward: -0.0036460052953662626 Last investment: [4.0209017]
Episode: 2850  Total Reward: -0.00358206273108387 Last investment: [4.071735]
Episode: 2900  Total Reward: -0.0035203251933365646 Last investment: [4.1238704]
Episode: 2950  Total Reward: -0.00346067888938892 Last investment: [4.175041]
Episode: 3000  Total Reward: -0.0034030201349144603 Last investment: [4.2256985]
Episode: 3050  Total Reward: -0.004362463097436797 Last investment: [4.2154984]
Episode: 3100  Total Reward: -0.004292123482509046 Last investment: [4.180911]
Episode: 3150  Total Reward: -0.00422401724694447 Last investment: [4.1475234]
Episode: 3200  Total Reward: -0.00415803760117956 Last investment: [4.1108894]
Episode: 3250  Total Reward: -0.004094698012396338 Last investment: [4.077522]
Episode: 3300  Total Reward: -0.004032676665278791 Last investment: [4.040059]
Episode: 3350  Total Reward: -0.003972505426983842 Last investment: [4.000724]
Episode: 3400  Total Reward: -0.0039144207339264405 Last investment: [3.9633963]
Episode: 3450  Total Reward: -0.0038579551208184513 Last investment: [3.9311175]
Episode: 3500  Total Reward: -0.003802857221787345 Last investment: [3.8960674]
Episode: 3550  Total Reward: -0.00374970861911026 Last investment: [3.8868942]
Episode: 3600  Total Reward: -0.0036978333372870795 Last investment: [3.8725922]
Episode: 3650  Total Reward: -0.0036473632179793154 Last investment: [3.8484962]
Episode: 3700  Total Reward: -0.003598088155904855 Last investment: [3.8214393]
Episode: 3750  Total Reward: -0.0035501264375814427 Last investment: [3.7910106]
Episode: 3800  Total Reward: -0.0035034265376840874 Last investment: [3.7592342]
Episode: 3850  Total Reward: -0.0034579395436340693 Last investment: [3.7268212]
Episode: 3900  Total Reward: -0.003413618561470909 Last investment: [3.6968195]
Episode: 3950  Total Reward: -0.0033704193203779355 Last investment: [3.6682508]
Episode: 4000  Total Reward: -0.003328299611163389 Last investment: [3.6371617]
Episode: 4050  Total Reward: -0.003287268582534186 Last investment: [3.6158514]
Episode: 4100  Total Reward: -0.0032471898540431966 Last investment: [3.6043906]
Episode: 4150  Total Reward: -0.0032081187330565194 Last investment: [3.602971]
Episode: 4200  Total Reward: -0.0031699359357733278 Last investment: [3.607753]
Episode: 4250  Total Reward: -0.003132651480599147 Last investment: [3.6125634]
Episode: 4300  Total Reward: -0.0030962339054113794 Last investment: [3.6193342]
Episode: 4350  Total Reward: -0.0030606531894318656 Last investment: [3.628889]
Episode: 4400  Total Reward: -0.0030259270724018856 Last investment: [3.6447132]
Episode: 4450  Total Reward: -0.0029919355312086047 Last investment: [3.6666446]
Episode: 4500  Total Reward: -0.002958699189380283 Last investment: [3.6924272]
Episode: 4550  Total Reward: -0.0029261931570279277 Last investment: [3.720241]
Episode: 4600  Total Reward: -0.0028944697870659643 Last investment: [3.7481291]
Episode: 4650  Total Reward: -0.002863353148279853 Last investment: [3.7865615]
Episode: 4700  Total Reward: -0.002832898425346384 Last investment: [3.8373432]
Episode: 4750  Total Reward: -0.0028030847184995983 Last investment: [3.8876917]
Episode: 4800  Total Reward: -0.0027738920017423663 Last investment: [3.9375782]
Episode: 4850  Total Reward: -0.002745301424138884 Last investment: [3.987184]
Episode: 4900  Total Reward: -0.002717294260998098 Last investment: [4.0281644]
Episode: 4950  Total Reward: -0.002689852882745991 Last investment: [4.0671115]
Episode: 5000  Total Reward: -0.002662960788444653 Last investment: [4.0984206]
Episode: 5050  Total Reward: -0.0026366000654379617 Last investment: [4.1269035]
Episode: 5100  Total Reward: -0.0026107561134366035 Last investment: [4.1616488]
Episode: 5150  Total Reward: -0.0025859918419276436 Last investment: [4.193217]
Episode: 5200  Total Reward: -0.0025611313192542926 Last investment: [4.219657]
Episode: 5250  Total Reward: -0.0025367450045872624 Last investment: [4.2545776]
Episode: 5300  Total Reward: -0.002513596184298282 Last investment: [4.2989726]
Episode: 5350  Total Reward: -0.0024901110062268087 Last investment: [4.3321667]
Episode: 5400  Total Reward: -0.002467058696945029 Last investment: [4.332133]
Episode: 5450  Total Reward: -0.0024444292841357745 Last investment: [4.3321447]
Episode: 5500  Total Reward: -0.002422212277099988 Last investment: [4.332174]
Episode: 5550  Total Reward: -0.002400394478022745 Last investment: [4.3321624]
Episode: 5600  Total Reward: -0.002378968247482076 Last investment: [4.332167]
Episode: 5650  Total Reward: -0.0023579211781276635 Last investment: [4.332186]
```

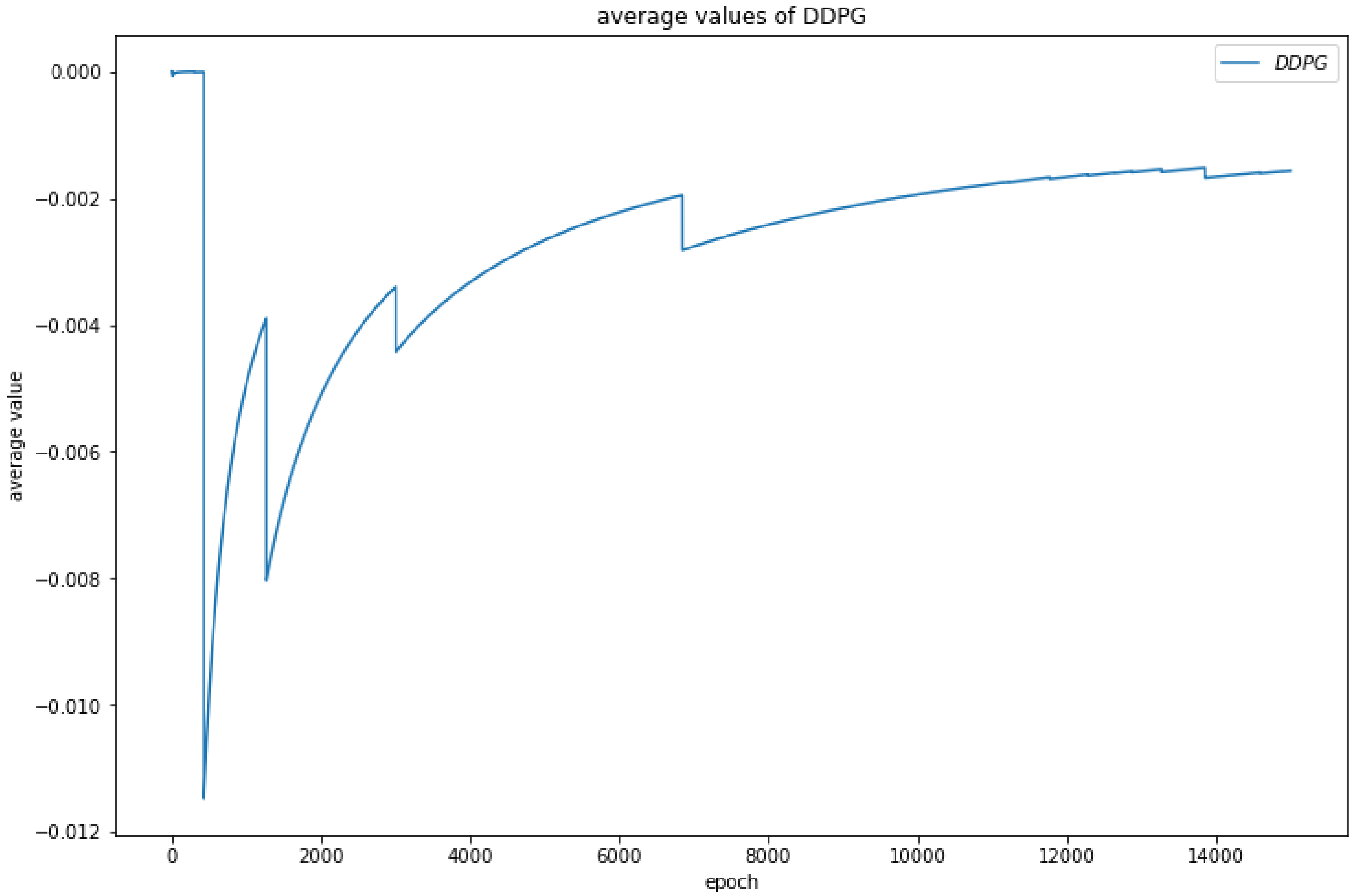
```
Episode: 5700  Total Reward: -0.0023372422884185935 Last investment: [4.332136]
Episode: 5750  Total Reward: -0.002316921980276415 Last investment: [4.3321805]
Episode: 5800  Total Reward: -0.0022969519618834507 Last investment: [4.3321614]
Episode: 5850  Total Reward: -0.002277323250129893 Last investment: [4.3321843]
Episode: 5900  Total Reward: -0.0022580271729675185 Last investment: [4.3322134]
Episode: 5950  Total Reward: -0.002239055347001964 Last investment: [4.3321815]
Episode: 6000  Total Reward: -0.0022203996645178033 Last investment: [4.332175]
Episode: 6050  Total Reward: -0.0022020522894556317 Last investment: [4.3321514]
Episode: 6100  Total Reward: -0.002184008447924777 Last investment: [4.3321676]
Episode: 6150  Total Reward: -0.002166255170208063 Last investment: [4.3321557]
Episode: 6200  Total Reward: -0.002148789108329774 Last investment: [4.332176]
Episode: 6250  Total Reward: -0.002131601548153968 Last investment: [4.332176]
Episode: 6300  Total Reward: -0.0021156136296741146 Last investment: [4.3321753]
Episode: 6350  Total Reward: -0.0020989578793494915 Last investment: [4.33217]
Episode: 6400  Total Reward: -0.0020834755701503857 Last investment: [4.3321667]
Episode: 6450  Total Reward: -0.0020673279870309976 Last investment: [4.3322]
Episode: 6500  Total Reward: -0.0020523262457476073 Last investment: [4.332185]
Episode: 6550  Total Reward: -0.0020366620255272035 Last investment: [4.3321586]
Episode: 6600  Total Reward: -0.002021235107708132 Last investment: [4.3321576]
Episode: 6650  Total Reward: -0.002006040139571785 Last investment: [4.3321714]
Episode: 6700  Total Reward: -0.0019910736286799595 Last investment: [4.3321953]
Episode: 6750  Total Reward: -0.0019771930572945705 Last investment: [4.3321605]
Episode: 6800  Total Reward: -0.0019626570131636473 Last investment: [4.3321457]
Episode: 6850  Total Reward: -0.0028212990608486475 Last investment: [4.332181]
Episode: 6900  Total Reward: -0.00280170580266254 Last investment: [4.3322306]
Episode: 6950  Total Reward: -0.0027815525463640352 Last investment: [4.3322024]
Episode: 7000  Total Reward: -0.002761688783653231 Last investment: [4.332191]
Episode: 7050  Total Reward: -0.00274210754442434 Last investment: [4.3321667]
Episode: 7100  Total Reward: -0.0027227996496941 Last investment: [4.3321996]
Episode: 7150  Total Reward: -0.0027053951851804966 Last investment: [4.332171]
Episode: 7200  Total Reward: -0.0026866111204580834 Last investment: [4.3321853]
Episode: 7250  Total Reward: -0.0026680853262596085 Last investment: [4.3321567]
Episode: 7300  Total Reward: -0.002649813276832905 Last investment: [4.331999]
Episode: 7350  Total Reward: -0.0026317905658115657 Last investment: [4.3321543]
Episode: 7400  Total Reward: -0.0026140136851875288 Last investment: [4.3321567]
Episode: 7450  Total Reward: -0.002596473856817117 Last investment: [4.332163]
Episode: 7500  Total Reward: -0.0025791671013421094 Last investment: [4.3321314]
Episode: 7550  Total Reward: -0.002562089544039092 Last investment: [4.332168]
Episode: 7600  Total Reward: -0.0025452359122980104 Last investment: [4.3321624]
Episode: 7650  Total Reward: -0.0025293673680910297 Last investment: [4.332177]
Episode: 7700  Total Reward: -0.0025129457787159905 Last investment: [4.3321853]
Episode: 7750  Total Reward: -0.002497488786401901 Last investment: [4.3321576]
Episode: 7800  Total Reward: -0.0024814827601931555 Last investment: [4.332192]
Episode: 7850  Total Reward: -0.0024656806043002267 Last investment: [4.332176]
Episode: 7900  Total Reward: -0.0024500777291857003 Last investment: [4.3321476]
Episode: 7950  Total Reward: -0.0024346710927391378 Last investment: [4.3321905]
Episode: 8000  Total Reward: -0.0024194570165535197 Last investment: [4.3321514]
Episode: 8050  Total Reward: -0.0024051587262327156 Last investment: [4.3321977]
Episode: 8100  Total Reward: -0.002390313903565739 Last investment: [4.332148]
Episode: 8150  Total Reward: -0.0023756512029304198 Last investment: [4.3321633]
Episode: 8200  Total Reward: -0.002361167988001159 Last investment: [4.332169]
Episode: 8250  Total Reward: -0.0023468603059424724 Last investment: [4.332177]
Episode: 8300  Total Reward: -0.002332724298405846 Last investment: [4.332168]
Episode: 8350  Total Reward: -0.0023187575641030254 Last investment: [4.332178]
Episode: 8400  Total Reward: -0.002304957080652941 Last investment: [4.3321557]
Episode: 8450  Total Reward: -0.0022913198965491726 Last investment: [4.3321834]
Episode: 8500  Total Reward: -0.002277845143708378 Last investment: [4.332127]
Episode: 8550  Total Reward: -0.0022645279732987993 Last investment: [4.332173]
Episode: 8600  Total Reward: -0.0022513636463201265 Last investment: [4.3321753]
Episode: 8650  Total Reward: -0.002238352810321364 Last investment: [4.3321557]
Episode: 8700  Total Reward: -0.0022261620236070844 Last investment: [4.3321733]
Episode: 8750  Total Reward: -0.0022134425520507774 Last investment: [4.3321724]
Episode: 8800  Total Reward: -0.0022008676035258743 Last investment: [4.332165]
Episode: 8850  Total Reward: -0.0021884353737293212 Last investment: [4.3321643]
Episode: 8900  Total Reward: -0.002176142175490897 Last investment: [4.332165]
Episode: 8950  Total Reward: -0.0021639869557011438 Last investment: [4.3321624]
```

```
Episode: 9000  Total Reward: -0.0021519667776747934 Last investment: [4.3321686]
Episode: 9050  Total Reward: -0.002140724027466654 Last investment: [4.332181]
Episode: 9100  Total Reward: -0.0021289631011721807 Last investment: [4.332167]
Episode: 9150  Total Reward: -0.0021173313206213082 Last investment: [4.3321643]
Episode: 9200  Total Reward: -0.00210582533704666 Last investment: [4.3321605]
Episode: 9250  Total Reward: -0.0020944437284379567 Last investment: [4.3321633]
Episode: 9300  Total Reward: -0.002083185104020672 Last investment: [4.3322167]
Episode: 9350  Total Reward: -0.002072046270431276 Last investment: [4.3321414]
Episode: 9400  Total Reward: -0.002061027134962118 Last investment: [4.332988]
Episode: 9450  Total Reward: -0.002050124605552159 Last investment: [4.335739]
Episode: 9500  Total Reward: -0.0020393368565494556 Last investment: [4.3498]
Episode: 9550  Total Reward: -0.0020286620989354754 Last investment: [4.3496647]
Episode: 9600  Total Reward: -0.002018706222770851 Last investment: [4.3394833]
Episode: 9650  Total Reward: -0.002008249539229976 Last investment: [4.35609]
Episode: 9700  Total Reward: -0.0019986109607058127 Last investment: [4.371875]
Episode: 9750  Total Reward: -0.001988364108875143 Last investment: [4.386733]
Episode: 9800  Total Reward: -0.0019782204316093547 Last investment: [4.4015126]
Episode: 9850  Total Reward: -0.001968179725438842 Last investment: [4.4153533]
Episode: 9900  Total Reward: -0.0019591413326124655 Last investment: [4.4286494]
Episode: 9950  Total Reward: -0.001950242636063162 Last investment: [4.4682574]
Episode: 10000  Total Reward: -0.0019404932859408209 Last investment: [4.5062947]
Episode: 10050  Total Reward: -0.001932242700615533 Last investment: [4.538549]
Episode: 10100  Total Reward: -0.001922679295960101 Last investment: [4.569666]
Episode: 10150  Total Reward: -0.0019132089064984137 Last investment: [4.593044]
Episode: 10200  Total Reward: -0.0019038326831693167 Last investment: [4.608028]
Episode: 10250  Total Reward: -0.0018967791966352542 Last investment: [4.626747]
Episode: 10300  Total Reward: -0.0018875724262568164 Last investment: [4.6481667]
Episode: 10350  Total Reward: -0.0018784546016522232 Last investment: [4.66648]
Episode: 10400  Total Reward: -0.0018724549017185644 Last investment: [4.6863904]
Episode: 10450  Total Reward: -0.0018634966474366968 Last investment: [4.6890507]
Episode: 10500  Total Reward: -0.001854627077251798 Last investment: [4.686107]
Episode: 10550  Total Reward: -0.0018458382150498501 Last investment: [4.6871834]
Episode: 10600  Total Reward: -0.0018371322541938235 Last investment: [4.6922765]
Episode: 10650  Total Reward: -0.0018285096930304304 Last investment: [4.685804]
Episode: 10700  Total Reward: -0.0018199660555257597 Last investment: [4.6881523]
Episode: 10750  Total Reward: -0.001814496419088484 Last investment: [4.689183]
Episode: 10800  Total Reward: -0.0018060967550331123 Last investment: [4.6937447]
Episode: 10850  Total Reward: -0.0017977778508779451 Last investment: [4.69502]
Episode: 10900  Total Reward: -0.0017895353432121586 Last investment: [4.712139]
Episode: 10950  Total Reward: -0.0017813646988735154 Last investment: [4.7133474]
Episode: 11000  Total Reward: -0.001773268325449289 Last investment: [4.7174716]
Episode: 11050  Total Reward: -0.0017652452166820609 Last investment: [4.7183757]
Episode: 11100  Total Reward: -0.0017572961430843562 Last investment: [4.7745814]
Episode: 11150  Total Reward: -0.0017544603865837967 Last investment: [4.814119]
Episode: 11200  Total Reward: -0.0017466310985165565 Last investment: [4.860687]
Episode: 11250  Total Reward: -0.0017460222568139905 Last investment: [4.901948]
Episode: 11300  Total Reward: -0.001738297180813896 Last investment: [4.9413815]
Episode: 11350  Total Reward: -0.001730643850817084 Last investment: [4.97601]
Episode: 11400  Total Reward: -0.001723062095484849 Last investment: [5.011184]
Episode: 11450  Total Reward: -0.001715538469897671 Last investment: [5.048374]
Episode: 11500  Total Reward: -0.0017080962285237905 Last investment: [5.087929]
Episode: 11550  Total Reward: -0.001700702519445659 Last investment: [5.120991]
Episode: 11600  Total Reward: -0.001693378872930095 Last investment: [5.145036]
Episode: 11650  Total Reward: -0.0016861185696442978 Last investment: [5.1659675]
Episode: 11700  Total Reward: -0.0016789135577810498 Last investment: [5.1666393]
Episode: 11750  Total Reward: -0.0016717768787509707 Last investment: [5.167535]
Episode: 11800  Total Reward: -0.001692038012658433 Last investment: [5.1690917]
Episode: 11850  Total Reward: -0.0016849202735875076 Last investment: [5.170617]
Episode: 11900  Total Reward: -0.0016778413804564218 Last investment: [5.1691747]
Episode: 11950  Total Reward: -0.0016708217128290287 Last investment: [5.165405]
Episode: 12000  Total Reward: -0.0016638741354460308 Last investment: [5.1603327]
Episode: 12050  Total Reward: -0.0016569706680187018 Last investment: [5.153785]
Episode: 12100  Total Reward: -0.0016501242510245497 Last investment: [5.144975]
Episode: 12150  Total Reward: -0.001643340291726898 Last investment: [5.133554]
Episode: 12200  Total Reward: -0.0016366118759788151 Last investment: [5.1190057]
Episode: 12250  Total Reward: -0.0016299323806517445 Last investment: [5.099447]
```

```
Episode: 12300  Total Reward: -0.0016407271165344486 Last investment: [5.073763]
Episode: 12350  Total Reward: -0.001634085035924678 Last investment: [5.042348]
Episode: 12400  Total Reward: -0.0016274965187408792 Last investment: [5.00564]
Episode: 12450  Total Reward: -0.001620968173298849 Last investment: [4.9645076]
Episode: 12500  Total Reward: -0.0016144880157238407 Last investment: [4.920092]
Episode: 12550  Total Reward: -0.0016080619160345126 Last investment: [4.886763]
Episode: 12600  Total Reward: -0.0016016812286049856 Last investment: [4.8668876]
Episode: 12650  Total Reward: -0.0016017986603359988 Last investment: [4.887021]
Episode: 12700  Total Reward: -0.0015954984476940393 Last investment: [4.928588]
Episode: 12750  Total Reward: -0.001589242084112679 Last investment: [4.960145]
Episode: 12800  Total Reward: -0.0015830345970341071 Last investment: [4.9942336]
Episode: 12850  Total Reward: -0.0015768834614913357 Last investment: [5.046422]
Episode: 12900  Total Reward: -0.0015858841031536578 Last investment: [5.0885677]
Episode: 12950  Total Reward: -0.0015797614757828494 Last investment: [5.126577]
Episode: 13000  Total Reward: -0.0015736919084785243 Last investment: [5.1639905]
Episode: 13050  Total Reward: -0.0015676694101914267 Last investment: [5.193075]
Episode: 13100  Total Reward: -0.0015617075100417706 Last investment: [5.221303]
Episode: 13150  Total Reward: -0.0015557699102041896 Last investment: [5.242779]
Episode: 13200  Total Reward: -0.0015498857806663477 Last investment: [5.262777]
Episode: 13250  Total Reward: -0.0015440376040085724 Last investment: [5.278577]
Episode: 13300  Total Reward: -0.0015805092510667784 Last investment: [5.318012]
Episode: 13350  Total Reward: -0.0015746107229921427 Last investment: [5.346904]
Episode: 13400  Total Reward: -0.0015687470632367332 Last investment: [5.373968]
Episode: 13450  Total Reward: -0.0015629157389385436 Last investment: [5.3985147]
Episode: 13500  Total Reward: -0.0015571412686781786 Last investment: [5.42471]
Episode: 13550  Total Reward: -0.0015514103542557283 Last investment: [5.4496336]
Episode: 13600  Total Reward: -0.0015457382855574436 Last investment: [5.4752326]
Episode: 13650  Total Reward: -0.0015400766631022736 Last investment: [5.5026975]
Episode: 13700  Total Reward: -0.0015344747670639567 Last investment: [5.532611]
Episode: 13750  Total Reward: -0.001528915510056206 Last investment: [5.5614347]
Episode: 13800  Total Reward: -0.001523398188945053 Last investment: [5.571312]
Episode: 13850  Total Reward: -0.0016786324761244183 Last investment: [5.5702863]
Episode: 13900  Total Reward: -0.0016725946670859166 Last investment: [5.567001]
Episode: 13950  Total Reward: -0.0016666001452938215 Last investment: [5.5592813]
Episode: 14000  Total Reward: -0.001660669066868008 Last investment: [5.546905]
Episode: 14050  Total Reward: -0.001654759649437686 Last investment: [5.5284786]
Episode: 14100  Total Reward: -0.0016489287414618433 Last investment: [5.5046263]
Episode: 14150  Total Reward: -0.001643118512104249 Last investment: [5.4744635]
Episode: 14200  Total Reward: -0.0016373332979614241 Last investment: [5.4390845]
Episode: 14250  Total Reward: -0.0016316141600874314 Last investment: [5.3998303]
Episode: 14300  Total Reward: -0.0016259215871157057 Last investment: [5.355707]
Episode: 14350  Total Reward: -0.0016202567575065853 Last investment: [5.3069997]
Episode: 14400  Total Reward: -0.0016146391078769994 Last investment: [5.2578816]
Episode: 14450  Total Reward: -0.0016090525112939292 Last investment: [5.2089267]
Episode: 14500  Total Reward: -0.0016035104409692059 Last investment: [5.160173]
Episode: 14550  Total Reward: -0.00159800568707264 Last investment: [5.1109905]
Episode: 14600  Total Reward: -0.0016059990707506292 Last investment: [5.0614905]
Episode: 14650  Total Reward: -0.0016005182240704113 Last investment: [5.0121565]
Episode: 14700  Total Reward: -0.0015950840903337703 Last investment: [4.963003]
Episode: 14750  Total Reward: -0.0015896802071334124 Last investment: [4.9139776]
Episode: 14800  Total Reward: -0.0015843193460806835 Last investment: [4.864598]
Episode: 14850  Total Reward: -0.0015789852983681628 Last investment: [4.8150167]
Episode: 14900  Total Reward: -0.0015768613049591594 Last investment: [4.765253]
Episode: 14950  Total Reward: -0.0015715878762043085 Last investment: [4.7154264]
```

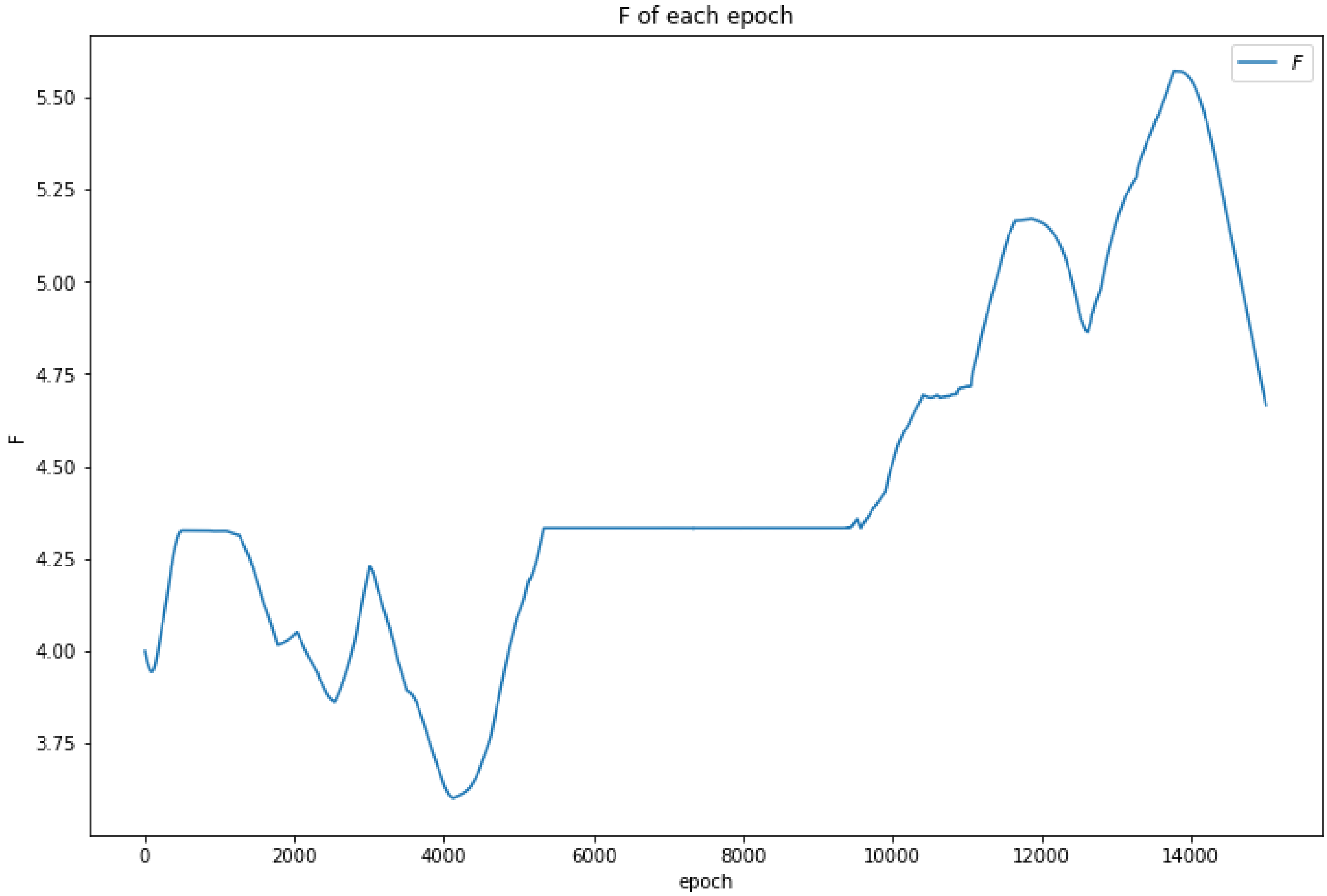
In [11]:

```python
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 8))
plt.plot(average_values,label='$DDPG$')
plt.xlabel('epoch')
plt.ylabel('average value')
plt.title('average values of DDPG')
plt.legend()
plt.show()
```



In [12]:

```
plt.figure(figsize=(12, 8))
plt.plot(last_investments,label='$F$')
plt.xlabel('epoch')
plt.ylabel('F')
plt.title('F of each epoch')
plt.legend()
plt.show()
```



2.4 Conclusion

The result shows DDPG can be used to solve this asset allocation problem. The average value of the problem using DDPG seems going to converge. However, we need to pay attention that the Policy Function (*Actor*) fluctuate a lot around optimal $F = 4.33217$, which could lead to the unstability of the whole system. It is because the huge randomness brought by multi-steps T, and a two-point distribution of Y_t. It is easy to generate wrong signal to update *Critic* and *Actor* in an improper way.