A Generalized Framework for Applications of DDPG in Portfolio Optimization

Ramsundar Govindarajan

Technische Universität München

April 18, 2023

Agenda

- Portfolio optimization problem
- DDPG
- DDPGFunctions
- DDPGShockBuffer
- DDPG Estimates
- Architecture
- Results
- Conclusion

Portfolio optimization problem

Assume, market model with 2 assets - risky(P_1) and riskless(P_0) - which follow the SDEs:

$$dP_1(t) = P_1(t)(\mu dt + \sigma dW(t)),$$

$$dP_0(t) = P_0(t)(r_c dt).$$

The portfolio optimization problem is defined as

$$(P) \begin{cases} \Phi(v_0) = \sup_{\pi \in \Lambda} \mathbb{E}[U(V^{v_0,\pi}(T))] \end{cases}$$

where the wealth update is defined by,

$$V^{v_0,\pi}(t) = v_0 \exp\left(\int_0^t r_c + (\mu - r_c)\pi(s, V^{v_0,\pi}(s)) - \frac{1}{2}(\sigma\pi(s, V^{v_0,\pi}(s)))^2 ds + \int_0^t \pi(s, V^{v_0,\pi}(s))\sigma dW(s) \right)$$

Goal: Maximize terminal value of utility function of wealth in a portfolio consisting of risky and riskless asset.

Portfolio optimization problem - Reinforcement learning

Traditional approaches are model specific and cannot be generalized
RL can be an alternative towards solving these problems

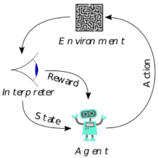


Figure: Reinforcement learning

Advantages of using RL based solution

- Flexibility : No assumptions on utility functions
- Model free: No specification for models such as Black-Scholes or Heston needed. Real market data can be used.

Solving Portfolio Optimization using RL - Q Learning

- Discretize problem
  Restrict set of admissible portfolio processes π to

$$\Lambda^{\Delta t} = \left\{ \pi^{\Delta t} = (\pi_i)_{i=0,\dots,(n-1)} | \pi_i = \pi(t_i, \cdot) : (0,\infty) \to \mathbb{R}, \pi^{\Delta t} \in \Lambda, i = 0, \dots, (n-1) \right\}.$$

  Define discretize version as $(P^{\Delta t}_{t_i}) \begin{cases} \Phi^{\Delta t}(t_i, v) = \sup\limits_{\pi \in \Lambda^{\Delta t}} \mathbb{E}[U(V^{v_0,\pi}(T)) | V^{v_0,\pi}(t_i) = v] \end{cases}$

- Define Q value function in the portfolio optimization context
  - Q value function represents expected future reward starting from state s, taking action a and acting optimally afterward
  - In portfolio optimization context,
    State s - wealth v_i at time t_i
    Action a - discretized relative portfolio processes $\Lambda^{\Delta t}$

Solving Portfolio Optimization using RL - Q Learning

- Use Bellman optimality equation and iteratively update Q value function by

$$Q(s,a) = \underset{s' \sim P}{\mathbb{E}}\left[r(s,a,s') + (1-d)\max_{a'} Q(s',a')\right]$$

- Compute optimal allocations by

$$a^*(s) = \underset{a \in A}{\operatorname{argmax}}\ Q(s,a).$$

DDPG

'Parametrized' Actor-Critic version of Q Learning

- ### Critic
  For a mini-batch of transitions $(s_i, a_i, r_i, s'i))$ from the replay buffer, the critic is updated by minimizing the loss function

$$L = \frac{1}{N}\sum_{i=1}^{N}(Q(s_i, a_i; \theta) - (r_i + \gamma * \max_a(Q'(s'_i, a; \theta')))^2$$

.

- ### Actor
  Updated as maximizer of the average Q-value

$$L = \frac{1}{N}\sum_{i=1}^{N} Q(s_i, a^{\phi}(s_i); \theta)$$

.

- ### Target networks
  Updated by a soft update

$$\theta' = \tau * \theta + (1 - \tau) * \theta'$$

Note: Q(s,a) action-value function, approximated by neural network with parameters θ.
$Q'(s,a)$ target action-value function
a^{ϕ} is the policy, approximated by neural network with parameters ϕ
$s_i = (V_i, t_i)$ is the current state, defined as a tuple of Wealth V at time t

$$r_i = \begin{cases} 0 & \text{when } t_{i+1} \neq T \\ U(V_{i+1}) & \text{Utility function of the evolution of wealth over time when } t_{i+1} = T \end{cases} \tag{1}$$

Problems with original implementation

- Model free or model generic approach leads to exploding runtimes
- Numerical instabilities
- Not scalable to complex problems

DDPGFunctions

Key Idea

Replace the neural networks in Critic and Actor function with general parametrized functions.

Example for power utility function,

- Critic function: Paramater $\boldsymbol{\theta} = (\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_T) \in \mathbb{R}^4$ and

$$Q(t_i, v, a; \boldsymbol{\theta}) = \frac{1}{b} v^b \exp\left((\boldsymbol{\theta}_0 + \boldsymbol{\theta}_1 a + \boldsymbol{\theta}_2 a^2)\Delta t + \boldsymbol{\theta}_T (T - t_{i+1}) \right),$$

- Actor function:

$$a(t, v; \boldsymbol{\phi}) = \boldsymbol{\pi}^* = \boldsymbol{\phi} \qquad \text{for some } \boldsymbol{\phi} \in \mathbb{R}$$

- Giving the functions a known structure and exploiting characteristics of final solution
- Faster convergence
- Appropriate proxies for Critic and Actor functions

DDPGFunction -Performance problems

Bellman optimality equation - Loss Minimization

For all states s,

$$0 = \left(\mathbb{E}[(r + \gamma \max_{a'} Q(s', a') - Q(s, a))]\right)^2,$$

$$0 = \mathbb{E}[(r + \gamma \max_{a'} Q(s', a') - Q(s, a))^2].$$

In classic DDPG setting,

$$L = \frac{1}{|B|} \sum_{(s,a,r,s') \sim B} [(r + \gamma \max_{a'} Q(s', a') - Q(s, a))^2].$$

- $Q(s', a')$ is only one realization among the many probable returns of the portfolio
- Unstable and can lead to incorrect results
- Exploding gradients problem

Key Idea

Consider a mini-batch B - log return of shocks.
Instead of one realization of s', consider pool of possible s' in minimizing the loss function

- Better approximation of Bellman loss
- Leads to stable updates

DDPG Shock Buffer

1. Consider $S = (V_i, t_i)$ take action $a = a(V_i, t_i)$
2. Observe log return of shock
$$\Delta P = (\mu - \frac{1}{2}\sigma^2)\Delta t + \sigma \Delta W$$
3. Store (V_i, t_i) in replay buffer R_a and ΔP in replay buffer R_p
4. Define the wealth update function
$$V^U(V_i, a, \Delta P) = V_i((1-a)r\Delta t + a\Delta P + \frac{1}{2}\sigma^2 a(1-a)\Delta t$$
5. For batches $B_a \subset R_a$ and $B_p \subset R_p$, (size m) update parameters θ of critic and action ϕ as
$$\theta \leftarrow \text{argmin}_{\theta'} \frac{1}{|B_a|}\Sigma(Q^{\theta'}(V_i, t_i, a) - \frac{1}{|B_p|}\Sigma(r(V^u, t_i) + 1_{(i \neq n)} Q^{\theta tgt}(V^u, t_{i+1}, a^{\phi}_{tgt}(V^u, t_{i+1}))^2$$
$$\phi \leftarrow \text{argmin}_{\phi} \frac{1}{|B_a|}\Sigma(Q^{\theta'}(V_i, t_i, a^{\phi'}(v_i, t_i)$$
Where V^u is the updated wealth generated from $(V_i, a, \Delta P)$
6. Update target network
7. Transition to $S = (V^U, t_i)$

DDPG Estimates

Key Idea

$$\mathbb{E}[(r + \gamma \max_{a'} Q(s', a') - Q(s, a))^2]$$

In Bellman optimality equation, observe Q(s',a') is function of a standard normal variate. Then we can discretize the integral and use numerical methods such as computing Riemann sum which can capture the expectation better.

- Leads to faster and better convergence
- Variance problem can be reduced.

DDPG Estimate - Algorithm

Replace "inner sum" in DDPG Shock Buffer by the following procedure

Finding better estimate of expectation

$$\begin{aligned} Q(v,t,a) &= \mathbb{E}[r(V^u(v,a,z),t)] + 1_{\{t+\Delta t \neq T\}}\mathbb{E}[\max_{a' \in A} Q(V^u(v,a,z),t+\Delta t,a')] \\ &= \int_{\mathbb{R}} r(V^u(v,a,z),t)\varphi(z)dz + 1_{\{t+\Delta t \neq T\}} \int_{\mathbb{R}} \max_{a' \in A} Q(V^u(v,a,z),t+\Delta t,a')\varphi(z)dz \\ &\approx \sum_{i=-(m-1)}^{m} [r(V^u(v,a,z_i),t) + 1_{\{t+\Delta t \neq T\}} \max_{a' \in A} Q(V^u(v,a,z_i),t+\Delta t,a')]\varphi(z_i)(z_i - z_{i-1}) \end{aligned}$$

1. For the mini-batch $B_a \subset R_a$, consider a sample $S = (V_i, t_i)$ and action a_i
2. Generate log return of shocks for "2m" standard normal variates z using the following equation.

$$\Delta P = (\mu - \frac{1}{2}\sigma^2)\Delta t + \sigma z\sqrt{\Delta t}$$

3. Define the wealth update function

$$V^U(V_i, a_i, \Delta P) = V_i((1-a).r.\Delta t + a_i \Delta P + \frac{1}{2}\sigma^2 a_i(1-a_i)\Delta t$$

4. Update parameters θ of critic and action ϕ as

$$\theta \leftarrow \operatorname{argmin}_{\theta'} \frac{1}{|B_a|} \Sigma(Q^{\theta'}(V_i, t_i, a) - 1_{i \neq n} Q(V^u, t_{i+1}, a^{\phi}_{tgt} V^u, t_{i+1}))^2$$

$$\phi \leftarrow \operatorname{argmin}_{\phi} \frac{1}{|B_a|} \Sigma(Q^{\theta'}(V_i, t_i, a^{\phi'}(v_i, t_i))$$

Architecture - Data flow diagram

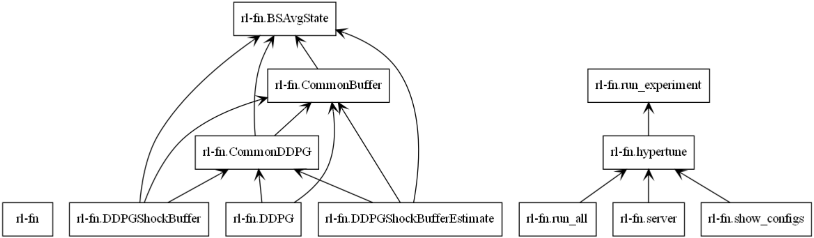
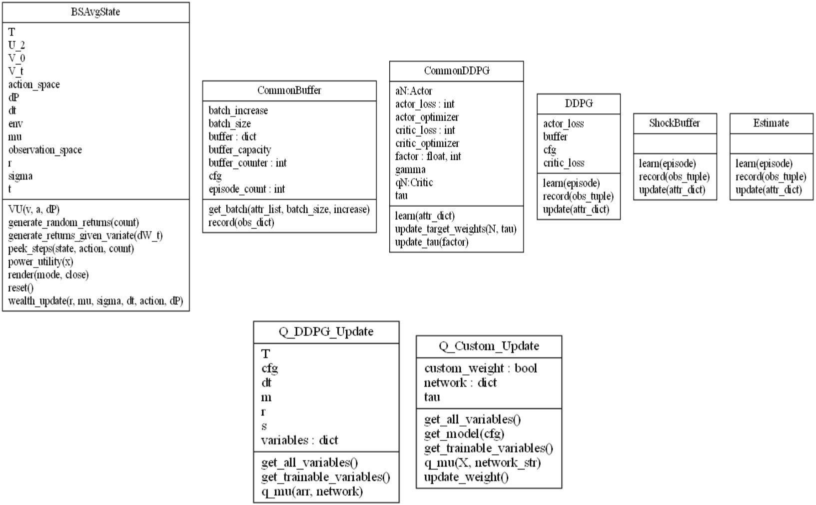


Figure: Data flow diagram

Architecture - Class diagrams



Architecture - Modular components

```
{
  "name": "Experiment - Decaying tau and batch size in DDPG Shock Buffer version",
  "env": {
    "name": "BlackScholes-V2",
    "mu": 0.09,
    "sigma": 0.3
  },
  "general_settings": {
    "max_episodes": 5000,
    "max_steps": 100,
    "batch_size": 1024,
    "batch_size_increase": "linear"
  },
  "ddpg": {
    "type" :
       {
      "name": "DDPGShockBufferEstimate",
       "m": 20
       },
    "gamma": 1,
    "noise_decay": 1
    "q": {
      "name": "q_pow_utparametric"
    },
    "a": {
      "name": "a_pow_ut1",
    }
  }
```

Architecture - Hypertuning framework

```
"tune": {
  "buffer.name": {
    "list": [ "DDPGShockBuffer","DDPG"   ]
  },
  "env.dt" :
  {
    "list" : [ 0.01,0.02,0.1,0.2]
  },
  "ddpg.max_episodes" :
  {
     "low" : 2000, "high" : 20000 , "step" : 100
  },

  "group":[
    {
      "env.mu": 0.019536,"env.sigma": 0.377183
    },
    {
      "env.b": -8.381621,"env.sigma": 0.57196
    }
  ],
  "group_2" :
  [
    {
    "ddpg.q.name" : "q_log_utparametric", "env.U_2" : "np.log"
    },
    {
      "ddpg.q.name" : "q_pow_utparametric","env.U_2" : "pow"
    } ]}
```

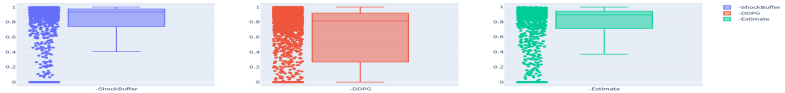
Architecture - Other components

1. MLFlow
2. Dashboard - https://ddpg-po-dash1.herokuapp.com/
3. Flask API deployment
4. Documentation service - https://rl-fn.readthedocs.io/en/latest/

Results - Configuration - All

Environment Parameter	Sampled values	DDPG Parameter	Values	DDPG Parameter	Values
$\mu \in [0,1]$	[0.07,0.955]	Version	DDPG, Shock Buffer, Estimates	Batch Size	1024
$\sigma \in [0,1]$	[0.1,1.4]	Grid Points	[8,1024] (8 - base case)	Batch Size Growth	None and Linear (Linear - base case)
Δt	[0.01,0.2] (0.2 - base case)	Shock Buffer Size	[8,1024] (8 - base case)		
$v_0 \in (0,1]$	[0.1,1] (1 base case)	Noise Decay	Linear and None		
Utility	power and log	Noise Scale	[0.1,5] (1 - base case)		
$b \in [-10,1)\setminus\{0\}$	[-9.0,0.95]	τ	5.10^{-4}		
T	1	τ decay	Linear and None (Linear - base case)		
r_c	0	Buffer Length	[10^4, 10^5] (10^4 - base case)		

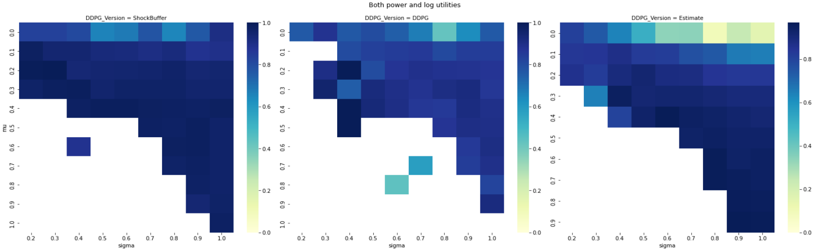
Results - All (accuracy)



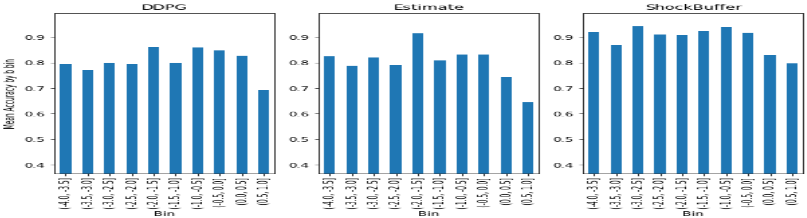
	25%	50%	75%	count	max	mean	min	std
DDPG	0.2705017919911987	0.8139643315095175	0.919626470577806	2278	1	0.6264779550229188	0	0.37804581820743255
Estimate	0.7127671413349081	0.8923287341451227	0.9426533015645605	1089	0.9971678571200332	0.7453510803262131	0	0.3086218198376786
ShockBuffer	0.7402616968297555	0.9339218505903728	0.9743309571024861	1483	1	0.7493961090812846	0	0.35886069370735263

Results - All

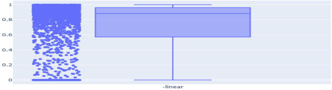
Accuracy - Environment



Accuracy - Risk aversion



τ decay and episodes analysis



	25%	50%	75%	count	max	mean	min	std
linear	0.5713752139464345	0.8824581732944529	0.9603886286570983	2801	1	0.6971292261512723	0	0.3655201831712523
none	0.5276514828746757	0.8633359990142171	0.9397194699814686	2049	1	0.6820398122671004	0	0.35856042664326465

Episodes

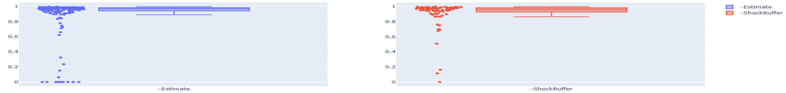
	25%	50%	75%	count	max	mean	min	std
20000	0.6662461030863227	0.843763583233776	0.902529742046045	7	0.998695082470896	0.7523285885961422	0.28628976420358854	0.2414107905918347
40000	0.8661955667218597	0.884375129232518	0.9480796336230617	7	0.9896142648008958	0.8909805210251462	0.7343238524527667	0.0848080235209624
80000	0.8345239745360862	0.9071918603680006	0.9678803141860244	6	0.9977351206634348	0.827304802744361	0.3532988303496992	0.24132738936720133

Robust configuration

Key observation

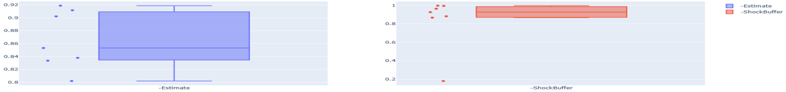
Observed that accuracy and convergence improved considerably by building better estimates of the expectation in Bellman equation. Construct better estimates by increasing the ”m” factor.

Robust configuration



	25%	50%	75%	count	max	mean	min	std
Estimate	0.9413537013902842	0.9741430518236585	0.9843286859341824	196	0.997020925411735	0.8788496589702375	0	0.26435173702663933
ShockBuffer	0.9340329959504001	0.9659764985511589	0.9861818212824847	81	0.9989960827570651	0.907655683504674	0	0.18174463698566953

Box plot Analysis M (timesteps - 0.01)



Conclusion

Parameter	Impact	DDPG Version
Stable estimate for expectation	High	Shock Buffer and Estimates
Number of grid points	High	Estimates
Batch size of log returns	High	Shock Buffer
Batch size of state, action tuples	Medium	All
τ	Medium	All
Noise scale	Low	All
Number of episodes	Medium	All
Model parameters	None	All
Time discretization Δt	High	All