第7章
深度确定性策略梯度方法

周炜星 谢文杰

华东理工大学金融学系

2023年秋

纲要

纲要

应用场景简介

- 复杂环境具有不可观察性、随机性、连续性、模型不可知等性质，我们需要建立合适的深度强化学习模型进行信息处理和智能决策。
- 在一些应用场景中，动作空间维度非常高且是连续变量，如在机器人领域，机器人手臂旋转的角度是连续变量。
- 机器人手臂又有很多自由度（动作空间维度），同时机器人的很多部位都必须活动，同样也会增加动作空间维度，构成了高维连续动作空间决策问题。
- 针对高维连续动作空间决策问题，即使我们将连续空间转变为离散化动作空间，使用DQN算法训练智能体也是异常困难的。

应用场景简介

- 连续动作空间可以使用确定性策略优化方法（Deterministic Policy Gradient）。
- 因此，为了适应高维状态空间和高维动作空间，深度确定性策略方法（Deep Deterministic Policy Gradient，DDPG）引入了深度神经网络模型来建模确定性策略函数和值函数。
- 众多深度强化学习算法各有优缺点，各类算法有对应的适用场景，并不存在一个最优的算法能够在所有的环境中都显著优于其它算法。
- 深度强化学习算法原理和神经网络结构在设计上也存在着交叉融合和互相学习，如深度确定性策略梯度算法（DDPG）和行动者-批评家（AC）算法具有同样的算法设计思想。

策略梯度方法比较

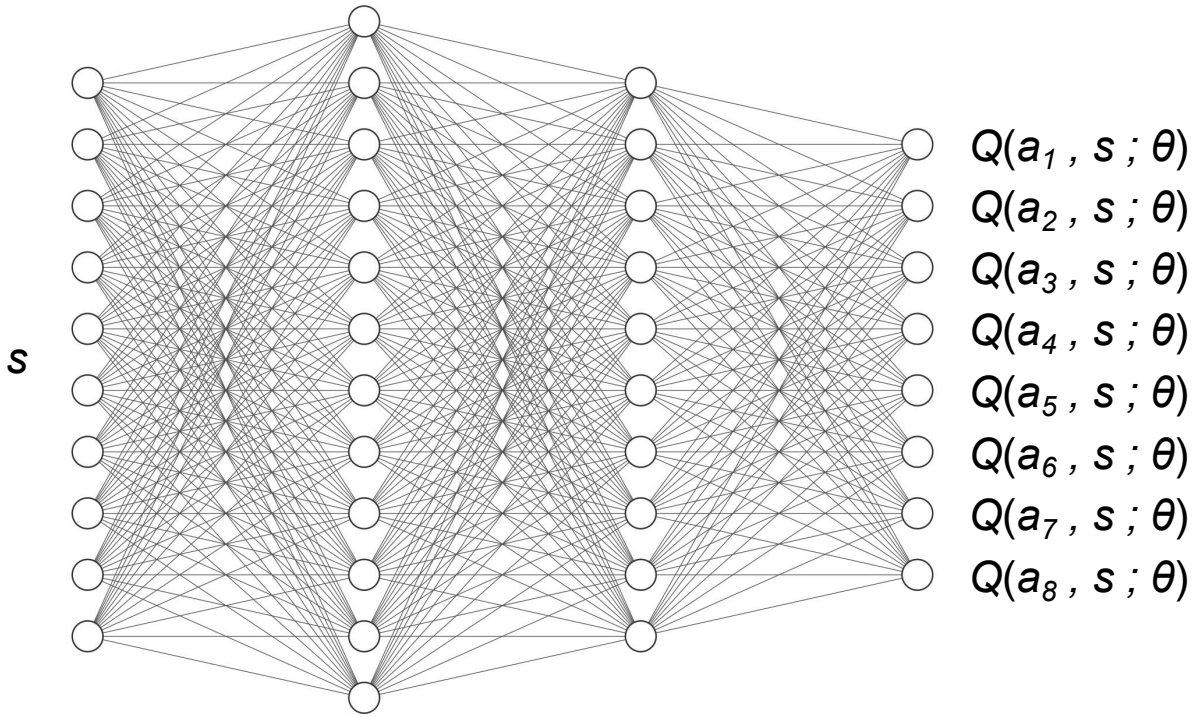


图 1: 值函数示意图

策略梯度方法比较

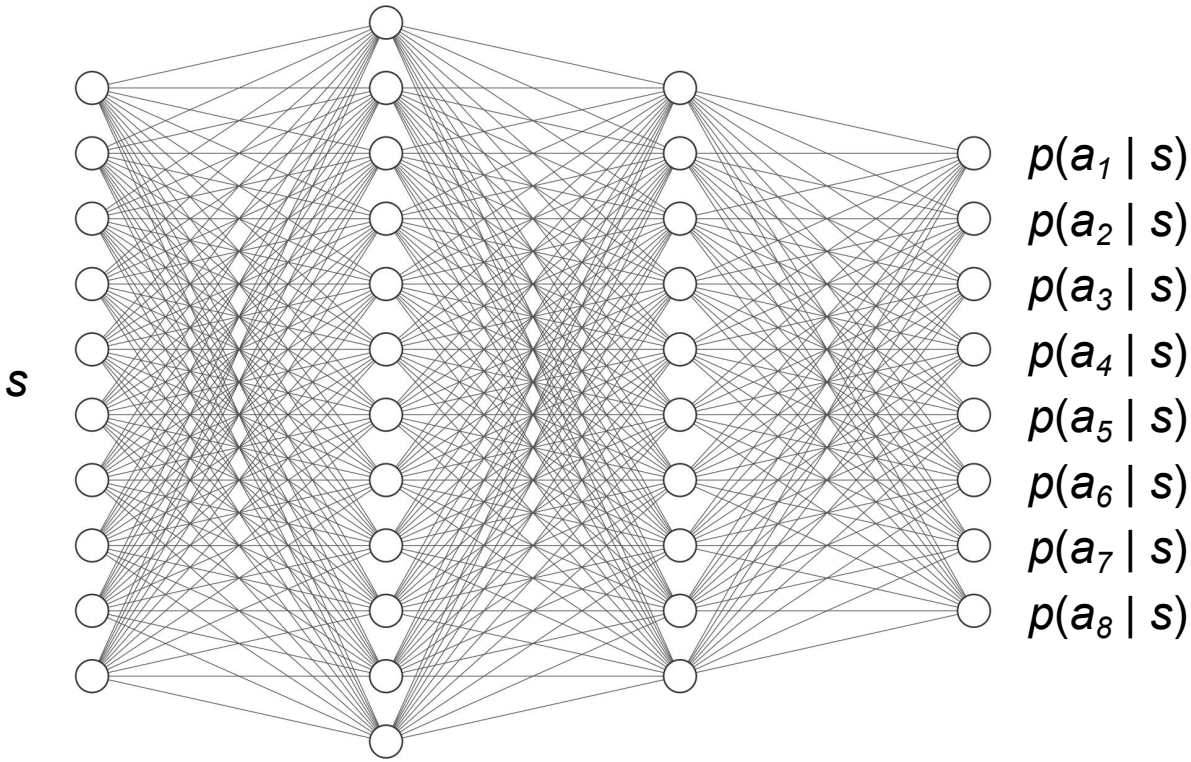


图 2: 随机性策略函数示意图

策略梯度方法比较

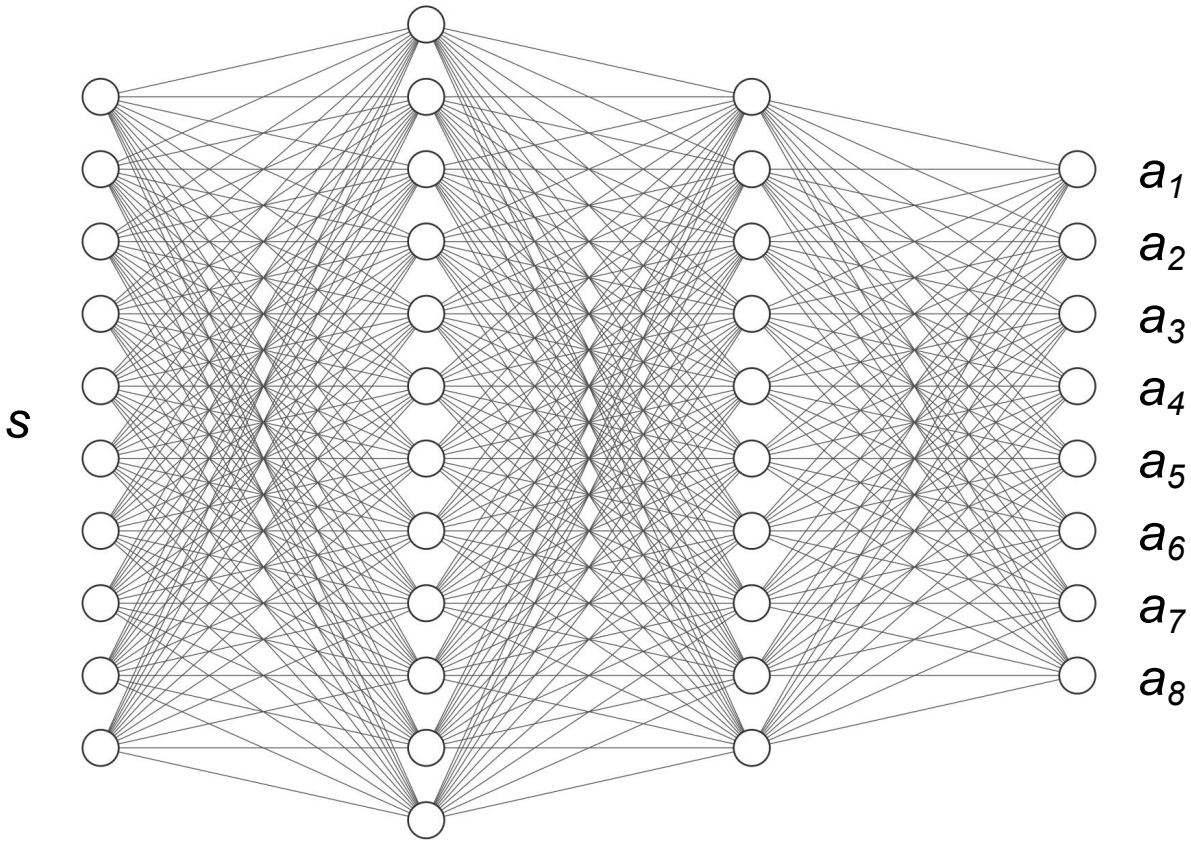


图 3: 确定性策略函数示意图

策略梯度方法比较

在DQN算法中，智能体在状态s下选择具有最大价值的动作a^*作为最优动作输出，隐式的策略函数为：

$$a^* = \arg\max_{a'} Q(s, a'; \theta). \tag{1}$$

随机性策略函数的输入参数为状态s，输出为8个动作的概率分布$p(a_i|s)$，随机性策略可以表示为：

$$p(a|s) = \pi_\theta(a|s). \tag{2}$$

其中，$\pi_\theta(a|s)$表示策略函数网络模型，θ为神经网络模型参数。确定性策略网络模型可以表示为：

$$a = \pi_\theta(s). \tag{3}$$

其中，$\pi_\theta(s)$表示确定性策略函数网络模型，θ为神经网络模型参数。

确定性策略函数的深度神经网络表示

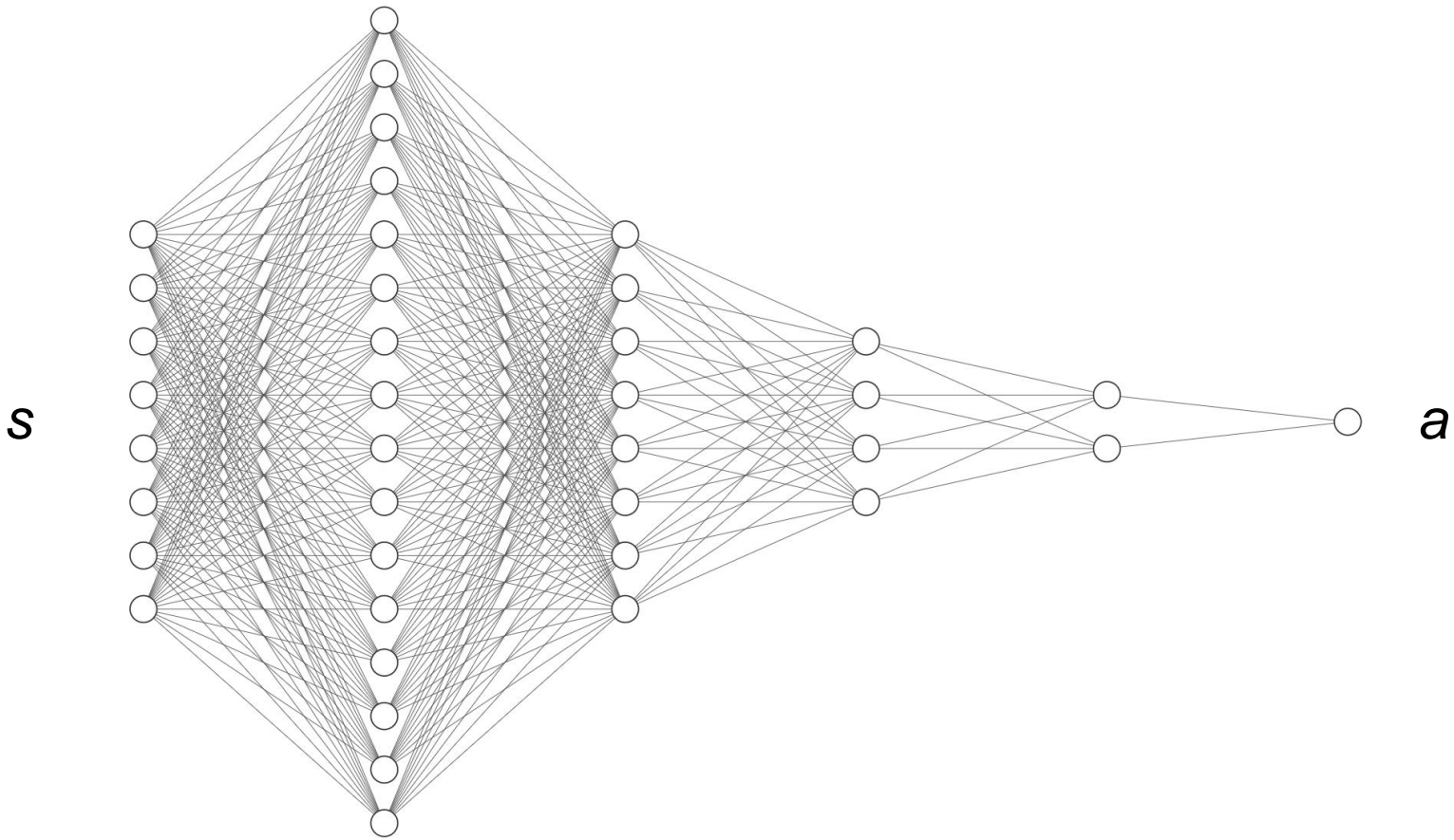


图 4: 确定性策略网络结构示意图

纲要

确定性策略梯度（Deterministic Policy Gradient）介绍

确定性策略函数为 $a = \pi(s)$，且

$$\pi(s) : \mathcal{S} \rightarrow \mathcal{A}, \tag{4}$$

确定性策略函数为状态空间到动作空间的映射。区别于随机性策略函数$\pi(s, a)$，确定性策略函数表示为

$$\pi(s, a) : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1], \tag{5}$$

其中，$\pi(s, a)$为状态空间和动作空间到实数域空间的映射。基于值函数的深度强化学习算法DQN可以简单地通过值函数获得最优化策略：

$$\pi_{k+1}(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ Q^{\pi_k}(s, a), \tag{6}$$

其中 π_k 是第 k 次迭代的策略函数。一般来说，DQN算法在具有离散动作空间的情景下应用较多。

确定性策略梯度定理

状态空间$\mathcal{S}$，动作空间$\mathcal{A}$，折扣系数γ，以及环境模型Env。初始化状态-动作值函数$Q(s,a;w)$参数w，初始化策略函数$\pi(s;\theta)$参数θ。最优化的目标函数设定为：

$$\begin{aligned} J(\pi_\theta) &= \int_{\mathcal{S}} \rho_\pi(s) r(s,\pi(s;\theta))\mathrm{ds} \\ &= \mathbb{E}_{s\sim\rho_{\pi_\theta}}\left[r(s,\pi(s;\theta))\right]. \end{aligned} \tag{7}$$

目标函数梯度为：

$$\begin{aligned} \nabla_\theta J(\pi_\theta) &= \int_{\mathcal{S}} \rho_\pi(s) \nabla_a Q(s,\pi(s;\theta);w)|_{a=\pi_\theta(s)} \nabla_\theta \pi_\theta(s)\mathrm{ds} \\ &= E_{s\sim\rho_\pi(s)} \nabla_a Q(s,\pi(s;\theta);w)|_{a=\pi_\theta(s)} \nabla_\theta \pi_\theta(s) \end{aligned} \tag{8}$$

目标函数

确定性策略梯度算法中的目标函数为：

$$\begin{aligned} J(\pi_\theta) &= \int_{\mathcal{S}} \rho_\pi(s) r(s, \pi(s;\theta)) \mathrm{ds} \\ &= \mathbb{E}_{s\sim\rho_{\pi_\theta}} \left[r(s, \pi(s;\theta)) \right]. \end{aligned} \tag{9}$$

根据确定性策略梯度定理可知，目标函数的梯度为：

$$\begin{aligned} \nabla_\theta J(\pi_\theta) &= \int_{\mathcal{S}} \rho_\pi(s) \nabla_a Q(s, \pi(s;\theta); w)|_{a=\pi_\theta(s)} \nabla_\theta \pi_\theta(s) \mathrm{ds} \\ &= E_{s\sim\rho_\pi(s)} \nabla_a Q(s, \pi(s;\theta); w)|_{a=\pi_\theta(s)} \nabla_\theta \pi_\theta(s) \end{aligned} \tag{10}$$

公式中的$r(s,a)$为回报函数，$\nabla_a \left(Q^{\pi_w}(s,a)\right)$ 为状态-动作值函数对动作a的梯度，$\nabla_\theta \pi_\theta(s)$为策略函数对参数$\theta$的梯度。

目标函数

公式推导时用到了函数导数的链式法则。公式中

$$Q(s,a;w) = Q(s,\pi_\theta(s);w), \tag{11}$$

状态-动作值函数$Q(s,\pi_\theta(s);w)$对参数θ求梯度时，将动作值函数参数w固定，动作值函数$Q(s,\pi_\theta(s);w)$看做是参数θ的函数。TensorFlow和Pytorch深度学习计算框架可以冻结住模型中的指定参数，只对需要求导的参数计算梯度值。因此，状态-动作值函数$Q(s,\pi_\theta(s);w)$对参数θ求导可得：

$$\begin{aligned}\nabla_\theta Q(s,\pi_\theta(s);w) &= \frac{\partial Q(s,a;w)}{\partial a}\frac{\partial a}{\partial \theta} \\ &= \frac{\partial Q(s,a;w)}{\partial a}\frac{\partial \pi_\theta(s)}{\partial \theta} \\ &= \nabla_a Q(s,a;w)\nabla_\theta \pi_\theta(s).\end{aligned} \tag{12}$$

目标函数

计算目标函数梯度后，可以采用梯度上升来更新策略函数参数θ，θ参数更新朝着动作值函数最大的方向变化，因此更新过程可简化为：

$$\theta = \theta + \alpha \nabla_a Q(s, \pi_\theta(s)) \nabla_\theta \pi_\theta(s), \tag{13}$$

其中α为学习率。
确定性策略梯度优化模型中的状态-动作值函数$Q(s, \pi_\theta(s); w)$的参数w同样需要迭代更新，其梯度算法与DQN算法类似。我们从经验池$\mathcal{D}$中随机采样数量为B的批量样本$\mathcal{B} = \{(s, a, r, s')\}$，计算TD目标值：

$$y = r(s, a) + \gamma \max_{a'} Q_{w^-}(s', \pi_{\theta^-}(s)). \tag{14}$$

目标函数

计算TD目标值：

$$y = r(s,a) + \gamma \max_{a'} Q_{w^-}(s', \pi_{\theta^-}(s)). \tag{15}$$

如果s'为终止状态，那么

$$y = r(s,a). \tag{16}$$

我们运用梯度下降算法更新状态-动作值函数$Q_w(s,a)$参数

$$w = w - \alpha_w \frac{1}{B} \nabla \sum_{(s,a,r,s') \sim \mathcal{B}} \left[(Q_w(s,a) - y))^2 \right]. \tag{17}$$

纲要

算法核心介绍

- 深度确定性策略梯度优化算法（Deep Deterministic Policy Gradient，DDPG）是一种非常经典的深度强化学习方法。
- DDPG算法基于确定性策略梯度定理，是一种离线策略优化算法，适用于连续动作空间中的复杂环境问题。
- DDPG算法与DNQ算法有诸多相似之处，被认为是针对连续动作空间的深度Q学习算法（DQN）的改进和优化。
- DDPG也采用Actor-Critic框架，算法同时会训练一个策略函数（Actor）和一个值函数（Critic）。

算法核心介绍

深度确定性策略梯度算法是DQN算法在连续动作空间的拓展，同样需要训练状态-动作值函数$Q(s,a)$，只是其中动作a属于连续动作空间$\mathcal{A}$，具体的Bellman方程表示如下：

$$Q(s,a)=\mathrm{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a)+\gamma\max_{a'}Q(s',a')\right], \tag{18}$$

其中，$r(s,a)$为智能体在状态s时执行动作a后得到的即时奖励，$s'\sim P(\cdot|s,a)$表示智能体在状态s时执行动作a后跳转到状态s'，$Q(s,a)$是智能体在状态s执行动作a获得的期望累积奖励。为了替换最大化操作$\max_{a'}Q(s',a')$，我们将通过其他方式最大化状态-动作值函数$Q(s,a)$：

$$Q(s,\pi_\theta(s))\approx\max_a Q(s,a). \tag{19}$$

算法核心介绍

确定性策略函数直接输出最优化动作是深度确定性策略梯度算法的核心。DDPG算法在进行迭代估计$Q(s,a)$过程中替换掉了$\max_{a'} Q(s',a')$最大化值函数过程，通过确定性策略函数$\pi_\theta(s)$直接输出最优化动作使得状态-动作值函数$Q(s,\pi_\theta(s))$最大。深度确定性策略梯度算法的关键之处在于，策略函数$\pi_\theta(s)$需要保证得到的动作能够使$Q(s,a)$最大。基于此思想，DDPG算法要求$Q(s,a)$对动作a是可微的，且策略函数$\pi_\theta(s)$也必须是可微的:

$$\nabla_\theta Q(s,\pi_\theta(s)) = \nabla_{\pi_\theta(s)} Q(s,\pi_\theta(s)).\nabla_\theta \pi_\theta(s) \tag{20}$$

策略函数$\pi_\theta(s)$的参数θ更新方向为状态-动作值函数$Q(s,\pi_\theta(s))$最大化方向。智能体策略函数按照梯度提升$\nabla_\theta Q(s,\pi_\theta(s))$进行更新，可以保证策略函数$\pi_\theta(s)$得到的动作使得$Q(s,a)$最大。因此，DDPG算法省略了$\max_{a'} Q(s',a')$最大化过程。

经验回放

DDPG算法中的经验数据用五元组(s, a, r, s', d)表示。在DDPG算法的经验数据中，d是状态s'是否为轨迹终点的标志变量。当s'为终止状态时，$d = 1$，否则$d = 0$。为了训练状态-动作值函数$Q_w(s, a)$，DQN算法中的TD目标值为：

$$y = r(s, a) + \gamma(1 - d) \max_{a'} Q(s', a'), \tag{21}$$

而DDPG算法中TD目标值为：

$$y = r(s, a) + \gamma(1 - d)Q(s', \pi_\theta(s')). \tag{22}$$

经验回放

因此，DDPG算法中的TD误差为：

$$\delta = (r(s,a) + \gamma(1-d)Q(s', \pi_\theta(s'))) - Q(s,a). \tag{23}$$

为了训练状态-动作值函数$Q_w(s,a)$，最小化TD误差，我们构建目标函数：

$$L(w, \mathcal{D}) = \mathrm{E}_{(s,a,r,s',d)\sim\mathcal{D}}\left[Q_w(s,a) - (r(s,a) + \gamma(1-d)Q(s', \pi_\theta(s')))\right]^2 \tag{24}$$

其中$\mathcal{D}$为经验池轨迹数据集。我们可以采用梯度下降算法更新参数w。

目标网络

为了增加训练稳定性，DDPG算法与DQN算法类似，设定了策略函数和状态-动作值函数的目标网络，分别记为$\pi_{\theta^-}(s)$和$Q_{w^-}(s,a)$。因此，状态-动作值函数的TD目标可表示为：

$$y = r(s,a) + \gamma(1-d)Q_{w^-}(s', \pi_{\theta^-}(s')). \tag{25}$$

相应的TD误差为：

$$Q_w(s,a) - (r(s,a) + \gamma(1-d)Q_{w^-}(s', \pi_{\theta^-}(s'))). \tag{26}$$

我们需要强调的是，此处计算TD目标值时使用策略函数目标网络进行估计，即π_{θ^-}输出最优动作。

目标网络

为了训练状态-动作值函数$Q_w(s,a)$，最小化TD误差，我们构建目标函数

$$L(w,\mathcal{D}) = \mathrm{E}_{(s,a,r,s',d)\sim\mathcal{D}}\left[\left(Q_w(s,a) - (r(s,a) + \gamma(1-d)Q_{w^-}(s',\pi_{\theta^-}(s')\right.\right. \tag{27}$$

其中$\mathcal{D}$为经验池数据集，我们采用梯度下降算法更新参数w，则状态-动作值函数$Q_w(s,a)$网络的参数更新如下：

$$w = w - \alpha_w \frac{1}{B}\nabla \sum_{(s,a,r,s',d)\sim\mathcal{B}}\left[\left(Q_w(s,a) - (r(s,a) + \gamma(1-d)Q_{w^-}(s',\pi_{\theta^-}\right.\right. \tag{28}$$

在迭代更新过程中，优化算法每次从经验池$\mathcal{D}$中采样数量为B的批量样本$\mathcal{B} = \{(s,a,r,s',d)\}$更新模型参数。

参数软更新

DDPG算法中目标网络的模型函数的参数更新采用软更新方法：

$$w^- = \rho w^- + (1-\rho)w. \tag{29}$$

DQN算法中设定了迭代C步以后更新目标网络参数$w^- = w$，也就是设定了$\rho = 0$。DDPG算法一般采用软更新策略，在一般情况下，ρ接近1。
DDPG算法也同样构造了一个与策略网络相同结构的目标网络，参数设为θ^-，用来替换$\max_{a'} Q(s', a')$的操作，策略函数目标网络函数的参数更新也采用软更新方法：

$$\theta^- = \rho\theta^- + (1-\rho)\theta, \tag{30}$$

一般情况下ρ设定为接近1。

参数软更新

策略函数更新较为简单，DDPG算法的策略函数优化目标不再是最大化累积收益，而是最大化状态-动作值函数$Q(s,a)$（本质上是一致的），使得策略函数$\pi_\theta(s)$得到的动作a最大化$Q(s,a)$。因此，策略函数$\pi_\theta(s)$的参数更新过程可简化为：

$$\theta = \theta + \alpha \nabla_{\pi_\theta(s)} Q(s, \pi_\theta(s)) \nabla_\theta \pi_\theta(s). \tag{31}$$

因为DDPG算法引入了目标网络技术，所以DDPG算法中有四个深度神经网络模型，分别为2个策略函数神经网络模型和2个状态-动作值函数神经网络模型。策略函数神经网络模型参数和状态-动作值函数神经网络模型参数采用常规梯度更新，各自对应的目标网络采用软更新方法。

深度确定性策略梯度算法伪代码

Algorithm 20: 深度确定性策略优化算法（DDPG）伪代码

Input: 状态空间 $\mathcal{S}$，动作空间 $\mathcal{A}$，折扣系数 γ 以及环境 Env

初始化状态-动作值函数 $Q_{\boldsymbol{w}}(s,a)$ 的参数 $\boldsymbol{w}$。初始化策略函数 $\pi_{\boldsymbol{\theta}}(s)$ 的参数 $\boldsymbol{\theta}$。同时初始化策略函数目标网络 $\pi_{\boldsymbol{\theta}^-}(s)$ 和状态-动作值函数目标网络 $Q_{\boldsymbol{w}^-}(s,a)$ 的参数：$\boldsymbol{w}^- = \boldsymbol{w}$ 和 $\boldsymbol{\theta}^- = \boldsymbol{\theta}$

Output: 最优策略函数 $a = \pi_{\boldsymbol{\theta}}(s)$

for $k = 0, 1, 2, 3, \cdots$ **do**

  智能体基于策略 $\pi_{\boldsymbol{\theta}}$ 与环境交互获得轨迹集合 $\mathcal{D}_k = \{\tau_i\}$

  经验池 $\mathcal{D}$ 中经验数据用五元组表示：(s, a, r, s', d)，s' 为终点时，$d = 1$，否则 $d = 0$。

  if 需要更新参数 **then**

    for $i = 0, 1, 2, \cdots$ **do**

      从经验池 $\mathcal{D}$ 中随机采样数量为 B 的小批量样本 $\mathcal{B} = \{(s, a, r, s', d)\}$

      计算 TD 目标值：

$$y = r(s,a) + \gamma(1-d)Q_{\boldsymbol{w}^-}(s', \pi_{\boldsymbol{\theta}^-}(s)) \tag{7.31}$$

      采用梯度下降算法更新状态-动作值函数参数：

$$\boldsymbol{w} = \boldsymbol{w} - \alpha_{\boldsymbol{w}} \frac{1}{B} \nabla \sum_{(s,a,r,s',d)\sim\mathcal{B}} \left[(Q_{\boldsymbol{w}}(s,a) - y)^2 \right] \tag{7.32}$$

      采用梯度上升更新策略函数参数：

$$\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \nabla_{\pi_{\boldsymbol{\theta}}(s)} Q(s, \pi_{\boldsymbol{\theta}}(s)) \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(s) \tag{7.33}$$

      采用软更新方法更新目标网络的参数：

$$\begin{aligned} \boldsymbol{w}^- &= \rho \boldsymbol{w}^- + (1-\rho)\boldsymbol{w} \\ \boldsymbol{\theta}^- &= \rho \boldsymbol{\theta}^- + (1-\rho)\boldsymbol{\theta} \end{aligned} \tag{7.34}$$

纲要

算法介绍

- DDPG算法在实际应用过程中表现出较好的智能决策性能，但DDPG算法在超参数调优方面具有脆弱性，算法性能受超参数影响较大。
- DDPG算法性能得益于深度神经网络模型和类似DQN算法模型架构，同时也继承了深度神经网络模型训练难这一与DQN模型相同的缺陷。
- DDPG算法与DQN算法具有非常类似的模型框架，因此也同样具有DQN算法类似的模型缺陷，如状态-动作值函数的过高估计等问题。
- 孪生延迟确定性策略梯度算法（Twin Delayed DDPG，TD3）改进了DDPG算法。TD3算法引入了孪生网络模型结构，主要改进可以从三方面进行分析。

孪生网络结构

孪生延迟确定性策略梯度算法（TD3）同时学习了两个状态-动作值函数$Q(s,a)$网络模型，分别为$Q_{w_1}(s,a)$和$Q_{w_2}(s,a)$，此处的孪生网络不同于DDPG算法中的目标网络。同样，TD3算法构建了两个与之对应的目标网络，分别为$Q_{w_1^-}(s,a)$和$Q_{w_2^-}(s,a)$。TD3算法中四个状态-动作值函数网络模型参数分别用w_1，w_2，w_1^-，w_2^-表示。因此，孪生延迟确定性策略梯度算法的TD目标值为：

$$y(r,s',d)=r(s,a)+\gamma(1-d)\min_{i=1,2}Q_{w_i^-}(s',\pi_{\theta^-}(s')). \tag{32}$$

需要强调的是，TD目标值中的“TD”表示时序差分学习（Temporal-Difference learning），TD3算法中“TD”表示Twin Delayed。TD3算法通过计算两个目标网络Q值，选择较小的Q值作为正式目标值，有效地减缓了过估计的风险。

孪生网络结构

TD3算法中的两个目标网络需要分别进行参数估计和优化，相应的目标函数分别为：

$$L(w_1, \mathcal{D}) = \mathrm{E}_{(s,a,r,s',d)\sim\mathcal{D}}\left[\left(Q_{w_1}(s,a) - y(r,s',d)\right)^2\right] \tag{33}$$

和

$$L(w_2, \mathcal{D}) = \mathrm{E}_{(s,a,r,s',d)\sim\mathcal{D}}\left[\left(Q_{w_2}(s,a) - y(r,s',d)\right)^2\right]. \tag{34}$$

孪生网络$Q_{w_1}(s,a)$和$Q_{w_2}(s,a)$同时估计动作值，通过选择两个动作值网络中的较小值作为目标值估计，一定程度上化解了Q函数过估计问题。

延迟参数更新

孪生延迟确定性策略梯度算法（Twin Delayed DDPG）中“Delayed”的含义为参数更新延迟。DDPG算法优化目标不再是直接最大化累积收益，而是最大化状态-动作值函数$Q(s,a)$，使得策略函数$\pi_\theta(s)$得到的动作能够使得$Q(s,a)$具有最大价值。因此，TD3算法中目标函数表示为：

$$\max_\theta E_{s\sim\mathcal{D}}\left[Q_{w_1}(s,\pi_\theta(s))\right]. \tag{35}$$

TD3至此都和DDPG处理方式一致，只是在更新频率方面做了调整。为了让策略函数计算的目标值更加稳定，减小TD目标值的波动性，TD3算法设置策略函数参数更新频率小于Q值函数参数更新频率，也就是说策略函数参数更新延迟于动作值函数参数更新。通常，我们可以设置状态-动作值函数$Q(s,a)$的参数更新频率是策略函数$\pi_\theta(s)$参数更新频率的2倍。

带裁剪的动作多样性改进

确定性策略梯度算法DDPG为了增加智能体的探索性能，在确定性策略函数输出的动作基础上增加了随机性：

$$a = \pi_\theta(s) + \epsilon, \tag{36}$$

其中ϵ为随机变量。一般来说，智能体的动作在动作空间中都有合法的动作区域，满足

$$a_{\text{Low}} \leq a \leq a_{\text{High}}. \tag{37}$$

在给确定性策略函数输出动作值添加随机性的过程中，也会进行合法值的验证，即对非法值进行了裁剪，表示为：

$$a = \text{clip}\left(\pi_\theta(s) + \epsilon, a_{\text{Low}}, a_{\text{High}}\right). \tag{38}$$

带裁剪的动作多样性改进

在高维空间中，DDPG算法的状态-动作值函数$Q(s,a)$通常具有无数尖峰值，这也是高维非线性函数的空间特性，具有较大波动性。如果状态-动作值函数$Q(s,a)$网络为某些动作生成了一个错误的尖峰，该策略将迅速利用该尖峰值，输出错误的动作，使得策略函数过早地收敛到错误策略。TD3算法通过裁剪类似的异常动作来避免此类情况的发生：

$$a = \text{clip}\left(\pi_\theta(s) + \text{clip}(\epsilon, -c, c), a_{\text{Low}}, a_{\text{High}}\right), \tag{39}$$

其中，c是一个超参数，限制随机变量产生的随机数的范围，且$\epsilon \sim \mathcal{N}(0,\sigma)$。在实际应用过程中，我们可以在动作的每个维度上添加剪切操作，目标动作被剪切到有效动作范围内。

TD3算法伪代码

Algorithm 21: 孪生延迟确定性策略梯度算法（Twin Delayed DDPG，TD3）伪代码

Input: 状态空间 $\mathcal{S}$，动作空间 $\mathcal{A}$，折扣系数 γ，以及环境 Env。初始化状态-动作值函数的孪生网络 $Q_{\boldsymbol{w}_1}(s,a)$ 的参数 $\boldsymbol{w}_1$ 和 $Q_{\boldsymbol{w}_2}(s,a)$ 的参数 $\boldsymbol{w}_2$，初始化策略函数 $\pi_{\boldsymbol{\theta}}(s)$ 的参数 $\boldsymbol{\theta}$，同时初始化策略函数目标网络和状态-动作值函数目标网络的参数：$\boldsymbol{w}_1^- = \boldsymbol{w}_1$、$\boldsymbol{w}_2^- = \boldsymbol{w}_2$ 和 $\boldsymbol{\theta}^- = \boldsymbol{\theta}$

Output: 最优策略函数 $\pi_{\boldsymbol{\theta}}(s)$

清空经验池 $\mathcal{D}$

for $k = 0, 1, 2, 3, \cdots$ **do**

  智能体基于策略 $\pi_{\boldsymbol{\theta}}$，获得动作：

$$a = \text{clip}\left(\pi_{\boldsymbol{\theta}}(s) + \text{clip}(\epsilon, -c, c), a_{\text{Low}}, a_{\text{High}}\right) \tag{7.45}$$

  将智能体与环境交互获得的轨迹存入经验池 $\mathcal{D} = \{\tau_i\}$，经验池 $\mathcal{D}$ 中的经验数据用五元组 (s, a, r, s', d) 表示，其中 d 作为判断状态 s' 是否为终止状态的标志，当 s' 为终点时，$d = 1$，否则 $d = 0$。

  if 需要更新参数 **then**

    for $j = 0, 1, 2, 3, \cdots$ **do**

      随机从经验池 $\mathcal{D}$ 中采样数量为 B 的小批量样本 $\mathcal{B} = \{(s, a, r, s', d)\}$

      计算目标行为动作：

$$a'(s') = \text{clip}\left(\pi_{\boldsymbol{\theta}^-}(s) + \text{clip}(\epsilon, -c, c), a_{\text{Low}}, a_{\text{High}}\right) \tag{7.46}$$

      计算 TD 目标值：

$$y(r, s', d) = r(s, a) + \gamma(1 - d) \min_{i=1,2} Q_{\boldsymbol{w}_i^-}(s', a'(s')) \tag{7.47}$$

      采用梯度下降算法更新孪生网络参数

      for $i = 1, 2$ **do**

$$\boldsymbol{w}_i = \boldsymbol{w}_i - \alpha_{\boldsymbol{w}} \frac{1}{B} \nabla \sum_{(s,a,r,s',d) \sim \mathcal{B}} \left[\left(Q_{\boldsymbol{w}_i}(s, a) - y(r, s', d) \right) \right)^2 \tag{7.48}$$

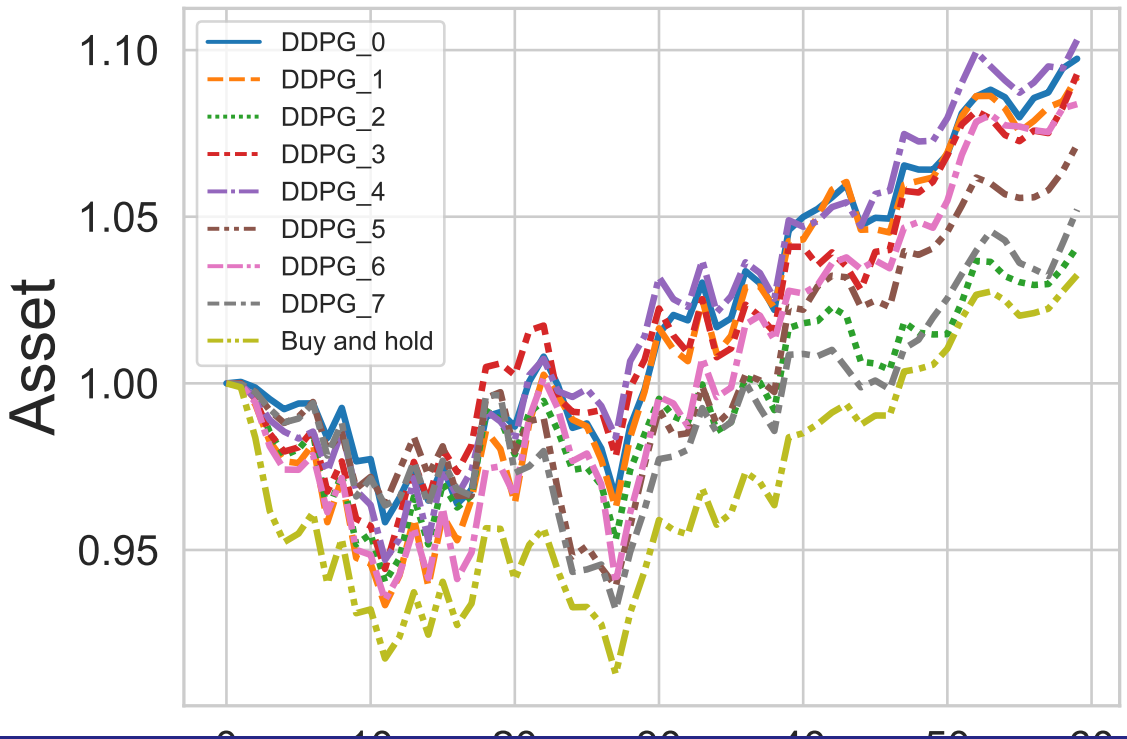
      if $j \bmod 2 == 0$ **then**

        采用梯度上升更新策略函数参数：

纲要

模型测试

与DQN、PPO2中的情况一样，为了保证模型测试的有效性，我们严格将测试集与训练集进行分离，训练集为2010年01月01日至2015年12月31日价格时间序，测试集为2016年01月01日至2016年03月31日价格时间序列。



课后习题

1. 什么是确定性策略？
2. 随机性策略函数的优势有哪些？
3. 确定性策略函数的优势有哪些？
4. 确定性策略的应用场景有哪些？
5. 确定性策略梯度定理与随机性策略梯度定理的差异？
6. DDPG和DQN的区别是什么？
7. TD3和DDPG的区别是什么？
8. 试辨析TD3和TD值中的“TD”的含义。
9. 运用确定性策略梯度优化算法实现一个智能交易强化学习系统。