第6章：深度策略优化方法

周炜星 谢文杰

华东理工大学金融学系

2023年秋

纲要

纲要

DQN的局限

- 深度强化学习算法的主要任务是训练智能体的策略函数。
- 在经典DQN算法中，智能体学习状态-动作值函数$Q(s,a;\theta)$，不是智能体的策略函数，或者说不是智能体策略函数的直接表示形式。
- 在面对连续动作空间时，取最大值操作的可行性较低且效率较低。
- 在经典DQN算法中，取最大值操作不够“soft”。虽然DQN算法采用$\epsilon-$贪心策略增加智能体探索能力，但面对复杂系统环境时仍然远远不够，存在较大的改进空间。
- 经典DQN算法的状态-动作值$Q(s,a;\theta)$表示在随机性策略方面存在局限。

策略函数的表示

- 深度强化学习的主要任务是学习智能策略，是否可以直接在经验数据集上优化策略函数？答案是肯定的。
- 策略函数的表示形式包括深度前馈神经网络、卷积神经网络、循环神经网络、图神经网络等。
- 图1是深度前馈神经网络模型的随机性策略函数示意图。

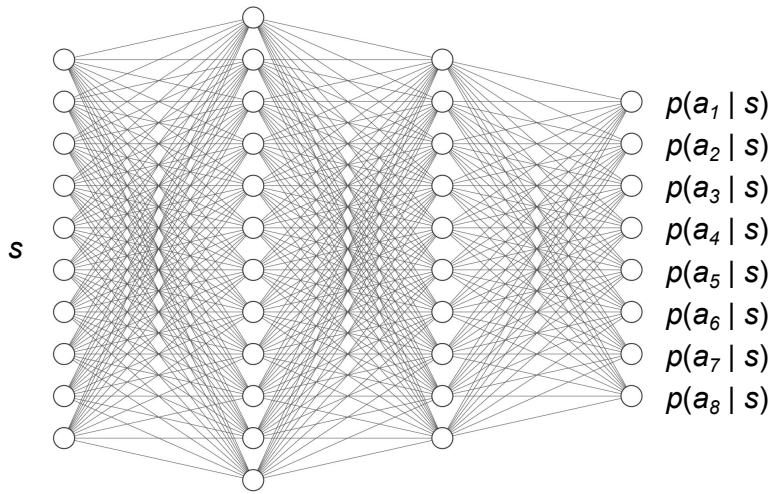


图 1: 深度前馈神经网络模型的随机性策略函数示意图

随机性策略函数

在图1中，深度神经网络模型的输入变量为智能体所处环境状态s，输出为动作空间中8个动作的概率分布$p(a_i|s)$，其中$i \in \{1, 2, 3, \cdots, 8\}$。智能体基于动作概率分布进行随机采样，就能够得到随机性动作。一般地，随机性策略可以表示为：

$$p(a|s) = \pi_\theta(a|s;\theta). \tag{1}$$

智能体基于策略函数的动作概率分布随机采样动作，输出动作具有随机性，同时也增加了动作的多样性，增强了智能体探索能力，提高了模型学习效率。

确定性策略函数

在图2中，确定性策略函数的神经网络模型的输入为状态s，输出为动作空间中的8个维度的动作a_i，其中$i \in \{1, 2, 3, \cdots, 8\}$。

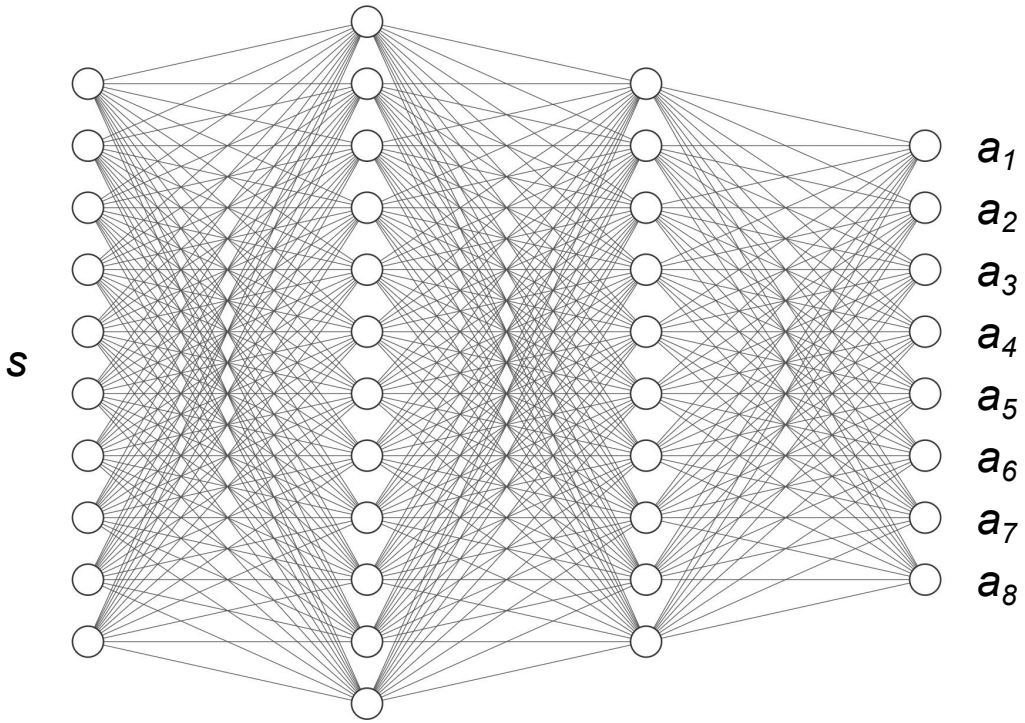


图 2: 确定性策略函数示意图

确定性策略函数

确定性策略函数的输出值不是概率值，无需随机采样动作空间，适用于连续动作空间。确定性策略函数输出动作a_i（$i \in \{1, 2, 3, \cdots, 8\}$）与随机性策略不一样。图1中随机性策略函数的 8个动作是8个离散动作，比如游戏动作空间中向上移动（a_1）、向下移动（a_2）、不动（a_3）等8个动作；而图2中确定性策略函数的输出动作a_i（$i \in \{1, 2, 3, \cdots, 8\}$）是指8个动作空间维度中的8个连续动作，比如机器人控制中的移动速度（a_1）、角度（a_2）、角速度（a_3）等。一般地，确定性策略函数可以表示成：

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

随机性策略梯度算法和确定性策略梯度算法各有优缺点，各有适用的实际场景。

轨迹数据

在深度强化学习中，智能体与环境交互获得经验数据，保存为经验轨迹数据τ，轨迹数据包含了智能体动作、环境状态、即时奖励信息、下一个状态等，如 $\tau = \{s_i, a_i, r_i, s_{i+1}, \}$，
$i \in \{0, 1, \cdots, T\}$。轨迹数据拼接和展开后可表示为：

$$s_0, a_0, r_0, s_1, a_1, r_1, \cdots, s_T, a_T, r_T. \tag{3}$$

轨迹数据中的s_0表示环境模型初始状态，s_T表示环境模型终止状态，a_i表示智能体动作，r_i表示智能体在环境状态s_i下执行动作a_i后获得的即时奖励回报。

目标函数

智能体从轨迹τ中获得的累积回报为：

$$R(\tau) = \sum_{t=0}^{T} r_t. \tag{4}$$

我们考虑具有折扣系数$\gamma < 1$的累积回报：

$$R(\tau) = \sum_{t=0}^{T} \gamma^t r_t. \tag{5}$$

同时，我们考虑无限长时间的累积收益情况，满足：

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t < r_{\max} \sum_{t=0}^{\infty} \gamma^t = r_{\max} \frac{1}{1-\gamma}. \tag{6}$$

式中$r_{\max}$表示即时奖励值的最大值。

梯度计算

马尔科夫决策过程是深度强化学习模型的基础模型框架，智能体从初始状态s_0开始，基于策略函数$\pi_\theta(a_t|s_t)$与环境模型交互，获得轨迹数据τ的概率可以表示为：

$$\begin{aligned} p_\theta(\tau) &= p(s_0)\pi_\theta(a_0|s_0)p(s_1|s_0,a_0)\pi_\theta(a_1|s_1)p(s_2|s_1,a_1)\cdots \\ &= p(s_0)\prod_{t=0}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t,a_t), \end{aligned} \tag{7}$$

其中，$p(s_0)$表示初始状态s_0出现的概率；$\pi_\theta(a|s)$为智能体随机性策略函数，即智能体在环境状态s下做出动作a的概率，θ为策略函数模型参数；$p(s'|s,a)$为环境状态转移函数，即环境在状态s情况下接收到智能体动作a后跳转到新状态s'的概率。

梯度计算

对$p_\theta(\tau)$取对数，可得：

$$\begin{aligned}\log p_\theta(\tau) &= \log\left(p(s_0)\prod_{t=0}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t,a_t)\right),\\ &= \log\left(p(s_0)\prod_{t=0}^{T}\pi_\theta(a_t|s_t)\prod_{t=0}^{T}p(s_{t+1}|s_t,a_t)\right),\\ &= \log p(s_0)+\sum_{t=0}^{T}\log\pi_\theta(a_t|s_t)+\sum_{t=0}^{T}\log p(s_{t+1}|s_t,a_t).\end{aligned} \tag{8}$$

梯度计算

智能体与环境交互得到大量的轨迹数据τ以及累积收益$R(\tau)$，因此，智能体的期望累积收益可以表示为：

$$J(\theta) = \sum_{\tau} R(\tau) p_\theta(\tau). \tag{9}$$

深度强化学习的目标是找到最优化的策略函数参数θ，即最优策略π_θ，使得智能体所获得的期望累积收益$J(\theta)$ 最大。因此，深度强化学习问题可以形式化为优化问题：

$$\pi_\theta^* = \arg\max_{\pi_\theta} J(\theta) = \arg\max_{\pi_\theta} \sum_{\tau} R(\tau) p_\theta(\tau). \tag{10}$$

梯度计算

针对该优化问题，可采用策略梯度方法。直接对目标函数求梯度，可得：

$$
\begin{aligned}
\nabla J(\theta) &= \sum_{\tau} R(\tau) \nabla p_\theta(\tau) \\
&= \sum_{\tau} R(\tau) p_\theta(\tau) \nabla \log p_\theta(\tau) \\
&= E_{\tau \sim p_\theta(\tau)} \left[R(\tau) \nabla \log p_\theta(\tau) \right] \\
&= E_{\tau \sim p_\theta(\tau)} \left[R(\tau) \nabla \left(\sum_{t=0}^{T} \log \pi_\theta(a_t | s_t) \right) \right] \\
&= E_{\tau \sim p_\theta(\tau)} \left[R(\tau) \left(\sum_{t=0}^{T} \nabla \log \pi_\theta(a_t | s_t) \right) \right]
\end{aligned}
\tag{11}
$$

梯度计算

由于状态概率分布、环境状态转移概率与策略函数参数θ无关，因此它们的梯度均为0。因此，在公式推导过程中用到了这两个比较关键的等式：

$$\nabla\left(\log p(s_0)\right)=0, \tag{12}$$

和

$$\nabla\left(\sum_{t=1}^{T}\log p(s_{t+1}|s_t,a_t)\right)=0. \tag{13}$$

我们展开分析一些公式推导细节，深入理解公式(11)的结果。在推导过程中，目标函数梯度$\nabla J(\theta)$转化成了策略函数对数的梯度

$$\frac{\nabla p_\theta(\tau)}{p_\theta(\tau)}=\nabla\log p_\theta(\tau). \tag{14}$$

梯度计算

在策略梯度算法中，目标函数梯度公式(11)包含了两类参数更新公式。一类是基于整体轨迹层面的梯度：

$$\nabla J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[R(\tau) \nabla \log p_\theta(\tau) \right]. \tag{15}$$

另一类是基于单步轨迹的梯度：

$$\nabla J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[R(\tau) \left(\sum_{t=0}^{T} \nabla \log \pi_\theta(a_t | s_t) \right) \right]. \tag{16}$$

我们从整体轨迹的层面能够很好地理解策略梯度更新的实际含义。

更新策略

策略梯度算法基于目标函数梯度更新策略函数参数θ。深度强化学习模型在训练过程中最大化目标函数，即智能体的期望累积收益。因此，我们可采用梯度上升优化算法，迭代优化策略函数参数，直到得到最大化期望累积收益。策略梯度算法中的策略函数参数更新公式为：

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t), \tag{17}$$

其中，α为学习率，控制参数更新的步长。参数更新的步长太小会导致更新速度较慢，参数更新的步长太大会导致算法不稳定。简单的推导和计算可以发现，策略梯度算法将目标函数优化问题简化成计算策略函数梯度问题，而与环境转换函数等无关，这极大便利了算法编程实现。

策略梯度优化理论

策略梯度定理保证了算法的有效性，策略梯度算法在实际应用过程中会遇到很多问题，如方差较大、算法不稳定、收敛速度慢等问题。因此，可以优化策略梯度计算过程：

$$\begin{aligned}\nabla_\theta J(\theta) &\propto E_{s\sim\mu(s),a\sim\pi_\theta(s,a)}\nabla_\theta \log(\pi_\theta(a|s,\theta))Q(s,a)\\ &\propto E_{\tau\sim\pi_\theta}\nabla_\theta \log(\pi_\theta(a|s,\theta))Q(s,a).\end{aligned} \tag{18}$$

权重函数用Φ_t表示，策略梯度优化算法可以表示如下：

$$\nabla_\theta J(\theta) \propto E_{s\sim\mu(s),a\sim\pi_\theta(s,a)}\sum_{t=0}^{T}\nabla_\theta \log(\pi_\theta(a_t|s_t,\theta))\Phi_t, \tag{19}$$

其中，策略梯度定理中的权重函数表示为：

$$\Phi_t = Q(s,a). \tag{20}$$

完整轨迹的累积奖励回报

强化学习中的REINFORCE算法考虑了整条轨迹τ的累积收益，将其作为衡量单个动作优劣的权重值，权重函数Φ_t可以表示为：

$$\Phi_t = R(\tau) = \sum_{t'=0}^{T} r_{\pi_\theta}(s_{t'}, a_{t'}, s_{t'+1}). \tag{21}$$

针对完整轨迹：

$$s_0, a_0, r_0, s_1, a_1, r_1, \cdots, s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, \cdots, s_T, a_T, r_T, \tag{22}$$

权重函数Φ_t可以表示为

$$\Phi_t = \sum_{t=0}^{T} r_t. \tag{23}$$

部分轨迹的累积奖励回报

基于模型的马尔科夫性，动作a_t的累积收益与动作$\{a_0, a_1, \cdots, a_{t-1}\}$均无关，只与动作$a_{t+1}$、动作$a_{t+2}$等有关，如下所示：

$$s_0, a_0, r_0, s_1, a_1, r_1, \cdots, \boxed{s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, \cdots, s_T, a_T, r_T}, \tag{24}$$

轨迹中状态s_t和动作a_t的累积收益只和动作a_t的后续发生的动作的即时奖励回报有关。因此，为了更加准确地度量动作的期望累积收益，将权重函数Φ_t设定为：

$$\Phi_t = \sum_{t'=t}^{T} r_{\pi_\theta}(s_{t'}, a_{t'}, s_{t'+1}) = \sum_{t'=t}^{T} r_t. \tag{25}$$

上式表明，动作a_t之前产生的即时奖励回报不需要算在当前动作的累积收益之中。

常数基线函数

在策略梯度优化算法中，为了减少策略优化过程中梯度估计的方差，提高智能体训练过程的稳定性和高效性，可设定基线函数为常数b，并将权重函数Φ_t设定为累积收益函数减去基线函数：

$$\Phi_t = \sum_{t'=t}^{T} r_{\pi_\theta}(s_{t'}, a_{t'}, s_{t'+1}) - b. \tag{26}$$

在策略优化过程中，不同的状态和动作累积收益减去相同的基线函数b后，能够在一定程度上减少梯度估计的方差。

基于状态的基线函数

考虑到复杂环境状态价值的差异性，为了进一步减小策略函数优化过程中梯度估计的方差，我们可将基线函数设定为状态的函数，用$b(s_t)$表示。因此，权重函数Φ_t变为：

$$\Phi_t = \sum_{t'=t}^{T} r_{\pi_\theta}(s_{t'}, a_{t'}, s_{t'+1}) - b(s_t). \tag{27}$$

设计不同的基线函数能够得到性能各异的策略梯度优化算法。基线函数可以设计成多种形式，如固定值b或基于状态变化的函数值$b(s_t)$。在策略梯度优化算法的实现过程中，基线函数$b(s_t)$的具体数值可以通过多次采样进行估计，如估计常数b或函数值$b(s_t)$。

基于状态值函数的基线函数

在于计算过程中，基线函数$b(s_t)$是一个数，或是一个列表函数，每个状态s_t都对应一个数值$b(s_t)$，而状态值函数$V(s_t)$可以是一个深度神经网络模型函数，作为状态的累积收益的估计值。策略梯度优化算法将状态值函数$V(s_t)$作为基线函数可得到新的权重函数Φ_t：

$$\Phi_t = \sum_{t'=t}^{T} r_{\pi_\theta}(s_{t'}, a_{t'}, s_{t'+1}) - V(s_t). \tag{28}$$

一般而言，权重函数Φ_t由两部分组成：一部分为状态-动作值的估计值，即状态-动作的期望累积收益；另一部分为基线函数。状态-动作值决定了策略梯度更新的准确性和偏差，而基线函数能减少策略梯度估计的方差。

基于自举方法的梯度估计

我们将优化累积奖励收益的估计值。权重公式中的$\sum_{t'=t}^{T} r_{\pi_\theta}(s_{t'}, a_{t'}, s_{t'+1})$表示一次完整轨迹的部分累积收益的计算值，与蒙特卡罗采样一致，我们需要计算部分轨迹采样的累积收益值。基于完整轨迹的蒙特卡罗采样的策略函数在训练过程容易不稳定，具有较大方差。因此，我们可以借鉴自举方法和TD方法，用深度神经网络模型估计累积奖励收益，用$r_{\pi_\theta}(s_t, a_t, s_{t+1}) + V(s_{t+1})$表示。如此，可以将权重函数$\Phi_t$写作：

$$\Phi_t = [r_{\pi_\theta}(s_t, a_t, s_{t+1}) + V(s_{t+1})] - V(s_t). \tag{29}$$

基于优势函数的策略梯度优化

策略梯度定理用深度神经网络估计状态-动作值函数$Q_{\pi_\theta}(s_t, a_t)$，其权重函数Φ_t为$Q_{\pi_\theta}(s_t, a_t)$。为了减小方差，将基线函数设定为状态值函数$V(s_t)$，可得：

$$\Phi_t = A_{\pi_\theta}(s_t, a_t) = Q_{\pi_\theta}(s_t, a_t) - V_{\pi_\theta}(s_t). \tag{30}$$

上式中的$A_{\pi_\theta}(s_t, a_t)$为优势函数，与Dueling DQN算法中的优势函数类似。

深度策略梯度优化算法

在深度策略梯度优化算法的具体实现过程中，梯度计算公式记作：

$$\nabla_\theta J(\theta) = \mathrm{E}_{s_0\sim\mu(s_0),a\sim\pi_\theta(s,a)} \sum_{t=0}^{T} \nabla_\theta \log(\pi_\theta(a_t|s_t)) A_{\pi_\theta}(s_t, a_t), \quad (31)$$

其中，E表示数学期望，$\pi_\theta(s, a)$表示策略函数，参数θ表示策略函数模型参数，$\mu(s_0)$表示状态概率分布，$A_{\pi_\theta}(s_t, a_t)$表示优势函数，记做：

$$A(s_t, a_t) = R_t - V(s_t) = \sum_{t'=t}^{T} r_{\pi_\theta}(s_{t'}, a_{t'}, s_{t'+1}) - V(s_t). \quad (32)$$

上式中的$R_t = \sum_{t'=t}^{T} r_{\pi_\theta}(s_{t'}, a_{t'}, s_{t'+1})$为累积收益值，$V(s_t)$为状态值函数，此处作为基线函数，用于减小梯度估计的方差。

深度策略梯度优化算法

我们需要确定状态值函数$V(s_t)$来辅助更新策略函数参数，设定状态值函数的深度网络模型为$V_w(s_t)$，其中w为深度网络模型参数。因此，我们构建状态价值函数$V_w(s_t)$的损失函数：

$$J_V(w) = \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} (V_w(s_t) - R_t)^2. \tag{33}$$

上式中的$|\mathcal{D}_k|$为第k次迭代中参数更新数据集$\mathcal{D}_k$的大小，T为一条轨迹样本τ的长度。我们采用回归方法估计状态值函数的参数w，拟合状态价值函数$V_w(s_t)$和累积收益R_t，最小化状态价值函数$V_w(s_t)$的损失函数：

$$\min \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} (V_w(s_t) - R_t)^2, \tag{34}$$

更新状态价值函数

我们运用梯度下降算法更新状态价值函数$V_w(s_t)$的参数：

$$w_{k+1} = w_k - \alpha_w \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} (V_w(s_t) - R_t) \nabla_{w_k} V_w(s_t), \quad (35)$$

其中α_w为状态值函数学习率，优势函数$A_{\pi_{\theta_k}}(s_t, a_t) = R_t - V(s_t)$，然后计算策略梯度：

$$\nabla_{\theta_k} J(\theta_k) = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \nabla_{\theta_k} \log(\pi_{\theta_k}(a_t|s_t, \theta_k)) A_{\pi_{\theta_k}}(s_t, a_t). \quad (36)$$

最后，我们更新策略函数参数θ，更新公式为：

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\theta_k), \quad (37)$$

其中α为学习率，θ_k为第k迭代过程中策略函数的参数。

DPG算法伪代码

Algorithm 17: 深度策略梯度优化算法伪代码

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

初始化状态值函数 $V_{\boldsymbol{w}}(s)$ 的参数 $\boldsymbol{w}$

初始化策略函数 $\pi_{\boldsymbol{\theta}}(s,a)$ 的参数 $\boldsymbol{\theta}$

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

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

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

计算每一条轨迹中状态的累积收益：$R_t=\sum\limits_{t'=t} r_{\pi_{\boldsymbol{\theta}}}(s_{t'},a_{t'},s_{t'+1})$

计算优势函数值：$A(s_t,a_t)=R_t-V(s_t)=\sum\limits_{t'=t}^{T} r_{\pi_{\boldsymbol{\theta}}}(s_{t'},a_{t'},s_{t'+1})-V(s_t)$

计算梯度公式：

$$\nabla_{\boldsymbol{\theta}_k}J(\boldsymbol{\theta}_k)=\frac{1}{|\mathcal{D}_k|}\sum_{\tau\in\mathcal{D}_k}\sum_{t=0}^{T}\nabla_{\boldsymbol{\theta}_k}\log(\pi_{\boldsymbol{\theta}_k}(a_t|s_t))A_{\pi_{\boldsymbol{\theta}_k}}(s_t,a_t) \tag{6.47}$$

更新策略函数参数：

$$\boldsymbol{\theta}_{k+1}=\boldsymbol{\theta}_k+\alpha\nabla_{\boldsymbol{\theta}_k}J(\boldsymbol{\theta}_k) \tag{6.48}$$

拟合状态价值函数和累积收益，使用回归方法估计状态值函数参数 $\boldsymbol{w}$，最小化目标函数：

$$\min\frac{1}{|\mathcal{D}_k|T}\sum_{\tau\in\mathcal{D}_k}\sum_{t=0}^{T}(V_{\boldsymbol{w}}(s_t)-R_t)^2 \tag{6.49}$$

更新状态值函数参数：

$$\boldsymbol{w}_{k+1}=\boldsymbol{w}_k-\alpha_{\boldsymbol{w}}\frac{1}{|\mathcal{D}_k|T}\sum_{\tau\in\mathcal{D}_k}\sum_{t=0}^{T}(V_{\boldsymbol{w}_k}(s_t)-R_t)\nabla_w V_{\boldsymbol{w}_k}(s_t) \tag{6.50}$$

近端策略优化算法的核心理论

深度强化学习算法的目标是最大化智能体的期望累积收益，即目标函数J_θ，策略梯度优化算法在迭代优化过程中的关键步骤是计算目标函数梯度。在策略梯度优化算法中，基于完整轨迹的累积收益的目标函数为：

$$J_\theta = E_{\tau\sim\pi_\theta(\tau)}\pi_\theta(\tau)R(\tau), \tag{38}$$

其中$R(\tau)$为轨迹样本τ的累积收益。为了增加智能体的学习效率和稳定性，课用优势函数$A_\theta(s_t, a_t)$辅助策略函数更新，目标函数梯度可以重写为：

$$\nabla J_\theta = E_{(s_t,a_t)\sim\pi_\theta}\left[\nabla \log \pi_\theta\left(a_t, s_t\right) A_\theta\left(s_t, a_t\right)\right]. \tag{39}$$

近端策略优化算法的核心理论

通过重要性采样技术对智能体采样过程进行改进，策略梯度优化算法的目标函数梯度可以重写为：

$$\nabla J_\theta = E_{(s_t,a_t)\sim\pi_{\theta_k}}\left[\frac{\pi_\theta(s_t,a_t)}{\pi_{\theta_k}(s_t,a_t)}\nabla\log\pi_\theta(a_t,s_t)A_\theta(s_t,a_t)\right] \quad (40)$$

近端策略优化算法是基于KL散度进行有约束的目标函数优化，PPO2算法将目标函数公式转化为：

$$J_{\theta_k}(\theta)\approx\sum_{(s_t,a_t)\sim\pi_{\theta_k}}\min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}A_{\theta_k}(s_t,a_t),\right.$$
$$\left.\text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)},1-\varepsilon,1+\varepsilon\right)A_{\theta_k}(s_t,a_t)\right), \quad (41)$$

其中，min为取最小值函数，clip函数为裁剪函数。

近端策略优化算法的核心理论

clip函数可以表述为分段函数形式，具体形式为：

$$\text{clip}\left(\frac{\pi_\theta\left(a_t|s_t\right)}{\pi_{\theta_k}\left(a_t|s_t\right)}, 1-\varepsilon, 1+\varepsilon\right)=\begin{cases}1-\epsilon, & \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}<1-\epsilon, \\ 1+\epsilon, & \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}>1+\epsilon, \\ \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}, & \text{else.}\end{cases} \tag{42}$$

上式中的clip函数对策略函数的比值$\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}$进行了限定，使得比值的范围为$[1-\epsilon, 1+\epsilon]$。在实际应用中，我们一般取$\epsilon=0.2$。

近端策略优化算法的核心理论

当$A_{\theta_k}(s_t, a_t) > 0$时，动作的优势函数为正，我们需要鼓励更多的此类具有优势的动作，即提高动作选择概率$\pi_\theta(a_t, s_t)$。为了提高模型训练的稳定性，动作选择概率$\pi_\theta(a_t, s_t)$不能大幅增大，新的策略函数$\pi_\theta(a_t, s_t) > (1+\epsilon)\pi_{\theta_k}(a_t, s_t)$时需要进行限制，使得$\pi_\theta(a_t, s_t)$不超过$(1+\epsilon)\pi_{\theta_k}(a_t, s_t)$，因此需要最小值函数min，而原始目标函数则变成：

$$J_{\theta_k}(\theta) \approx \sum_{(s_t,a_t)\sim\pi_{\theta_k}} \min\left(\frac{\pi_\theta\left(a_t|s_t\right)}{\pi_{\theta_k}\left(a_t, s_t\right)}, 1+\varepsilon\right) A_{\theta_k}\left(s_t, a_t\right). \tag{43}$$

近端策略优化算法的核心理论

当$A_{\theta_k}(s_t, a_t) < 0$时，动作的优势函数为负，我们需要减小动作选择概率$\pi_\theta(a_t, s_t)$。为了提高模型训练的稳定性，动作选择概率$\pi_\theta(a_t, s_t)$不能大幅减小，新的策略函数$\pi_\theta(a_t, s_t) < (1-\epsilon)\pi_{\theta_k}(a_t, s_t)$时需要进行限制，使得$\pi_\theta(a_t, s_t)$不小于$(1-\epsilon)\pi_{\theta_k}(a_t, s_t)$。因此，我们需要一个最大值函数max，则新旧策略函数比值满足：

$$\max\left(\frac{\pi_\theta\left(a_t|s_t\right)}{\pi_{\theta_k}\left(a_t, s_t\right)}, 1-\varepsilon\right). \tag{44}$$

因为$A_{\theta_k}(s_t, a_t) < 0$，所以新旧策略函数比值满足：

$$\min\left(\left(\frac{\pi_\theta\left(a_t|s_t\right)}{\pi_{\theta_k}\left(a_t, s_t\right)}, 1-\varepsilon\right) A_{\theta_k}\left(s_t, a_t\right)\right). \tag{45}$$

近端策略优化算法的核心理论

目标函数可以表示成：

$$J_{\theta_k}(\theta) \approx \sum_{(s_t,a_t)\sim\pi_{\theta_k}} \min\left(\left(\frac{\pi_\theta\left(a_t|s_t\right)}{\pi_{\theta_k}\left(a_t,s_t\right)},1-\varepsilon\right)A_{\theta_k}\left(s_t,a_t\right)\right). \tag{46}$$

公式中$\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t,s_t)}>0$，即$\pi_\theta\left(a_t|s_t\right)$最大变化情况是将动作选择概率减小到0。最后，通过策略梯度更新参数：

$$\begin{aligned}\theta_k = \arg\max_\theta \sum_{(s_t,a_t)\sim\pi_{\theta_k}} &\min\left(\frac{\pi_\theta\left(a_t|s_t\right)}{\pi_{\theta_k}\left(a_t|s_t\right)}A_{\theta_k}\left(s_t,a_t\right),\right.\\ &\left.\operatorname{clip}\left(\frac{\pi_\theta\left(a_t|s_t\right)}{\pi_{\theta_k}\left(a_t|s_t\right)},1-\varepsilon,1+\varepsilon\right)A_{\theta_k}\left(s_t,a_t\right)\right).\end{aligned} \tag{47}$$

近端策略优化算法PPO2伪代码

Algorithm 19: 近端策略优化算法（PPO2）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

初始化状态值函数 $V_{\boldsymbol{w}}(s)$ 的参数 $\boldsymbol{w}$

初始化策略函数 $\pi_{\boldsymbol{\theta}}(s,a)$ 的参数 $\boldsymbol{\theta}$

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

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

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

计算每一条轨迹中状态的累积收益：$R_t=\sum\limits_{t'=t} R_{\pi_{\boldsymbol{\theta}}}(s_{t'},a_{t'},s_{t'+1})$

计算优势函数值：$A(s_t,a_t)=R_t-V(s_t)=\sum\limits_{t'=t}^{T} R_{\pi_{\boldsymbol{\theta}}}(s_{t'},a_{t'},s_{t'+1})-V(s_t)$

策略梯度优化算法更新参数：

$$\boldsymbol{\theta}_{k+1}=\arg\max_{\boldsymbol{\theta}}\sum_{(s_t,a_t)\sim\pi_{\boldsymbol{\theta}_k}}\min\left(\frac{\pi_{\boldsymbol{\theta}}(a_t|s_t)}{\pi_{\boldsymbol{\theta}_k}(a_t|s_t)}A_{\boldsymbol{\theta}_k}(s_t,a_t),\right.\\ \left.\operatorname{clip}\left(\frac{\pi_{\boldsymbol{\theta}}(a_t|s_t)}{\pi_{\boldsymbol{\theta}_k}(a_t|s_t)},1-\varepsilon,1+\varepsilon\right)A_{\boldsymbol{\theta}_k}(s_t,a_t)\right) \tag{6.79}$$

采用回归估计状态值函数参数 $\boldsymbol{w}$，拟合状态价值函数：

$$\min\frac{1}{|\mathcal{D}_k|T}\sum_{\tau\in\mathcal{D}_k}\sum_{t=0}^{T}(V_{\boldsymbol{w}}(s_t)-R_t)^2 \tag{6.80}$$

状态值函数参数更新公式：

$$\boldsymbol{w}_{k+1}=\boldsymbol{w}_k-\alpha_{\boldsymbol{w}}\frac{1}{|\mathcal{D}_k|T}\sum_{\tau\in\mathcal{D}_k}\sum_{t=0}^{T}(V_{\boldsymbol{w}}(s_t)-R_t)\nabla V_{\boldsymbol{w}}(s_t) \tag{6.81}$$

模型训练中智能体单个训练周期内的累积收益情况

图3给出了8个智能体单个训练周期内的累积收益情况，衡量智能交易系统中智能体投资决策的累积回报收益。

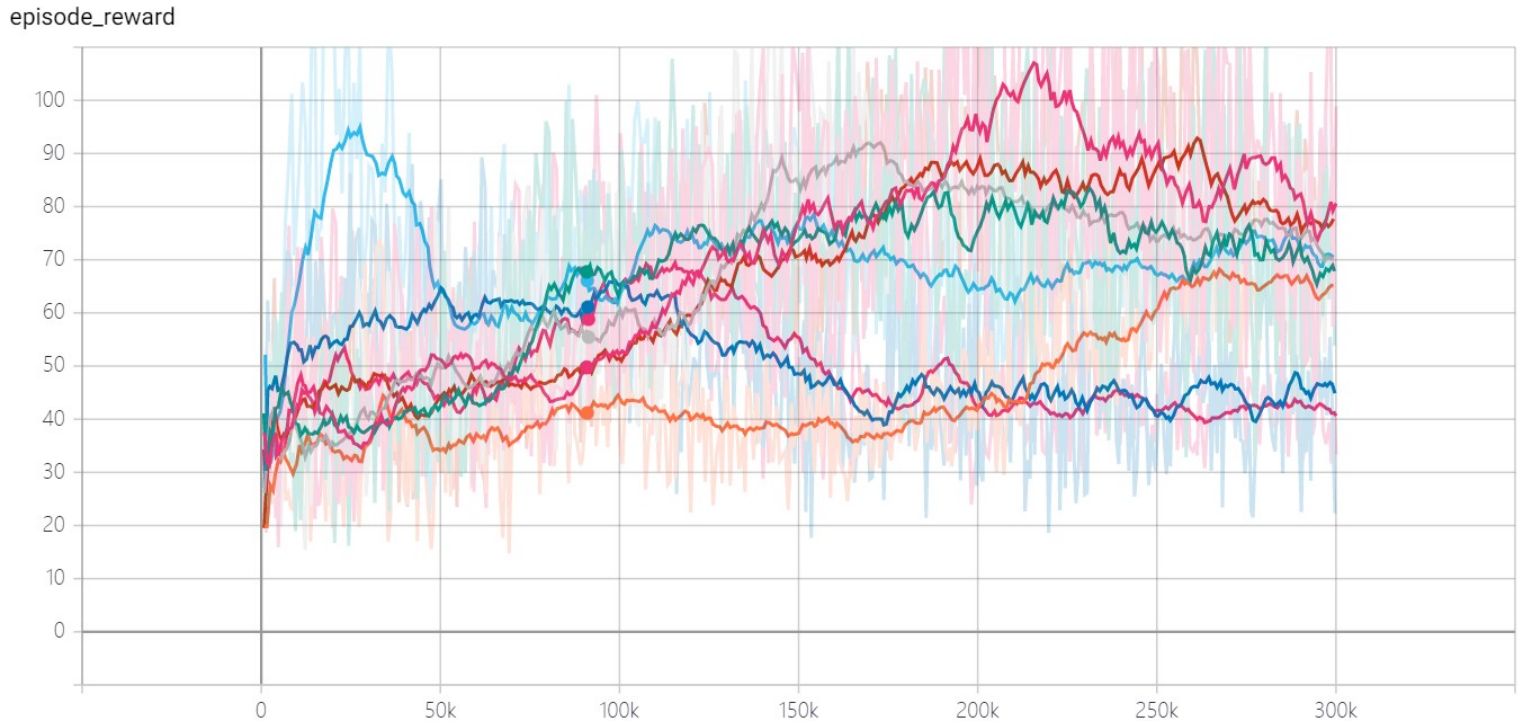


图 3: PPO2模型训练中智能体单个训练周期内的累积收益情况

模型测试

在金融市场环境建模时，我们在数据处理过程中将训练集和测试集进行了严格的划分，我们在测试集中测试智能体投资收益情况，结果如图4所示。

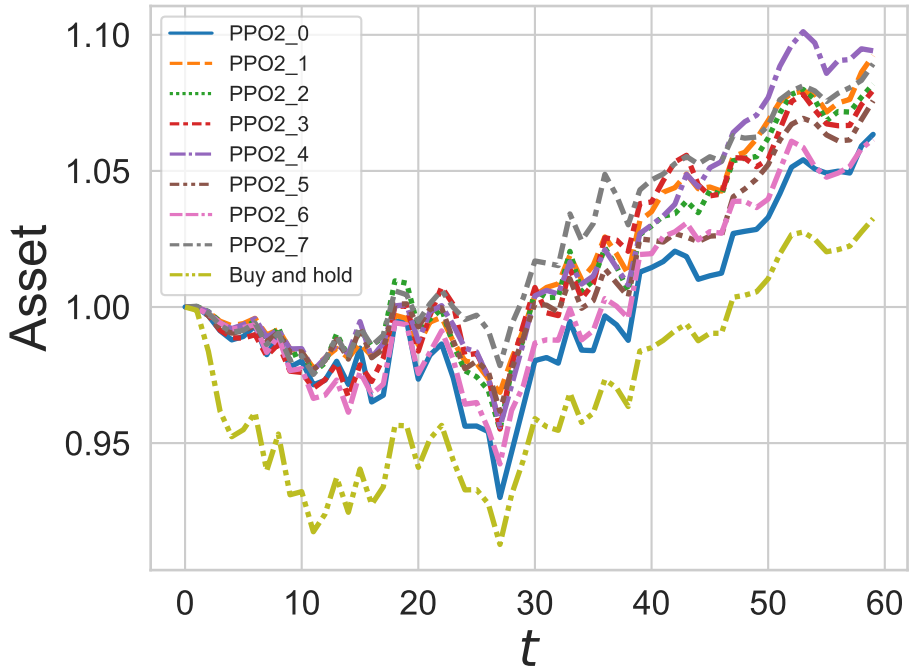
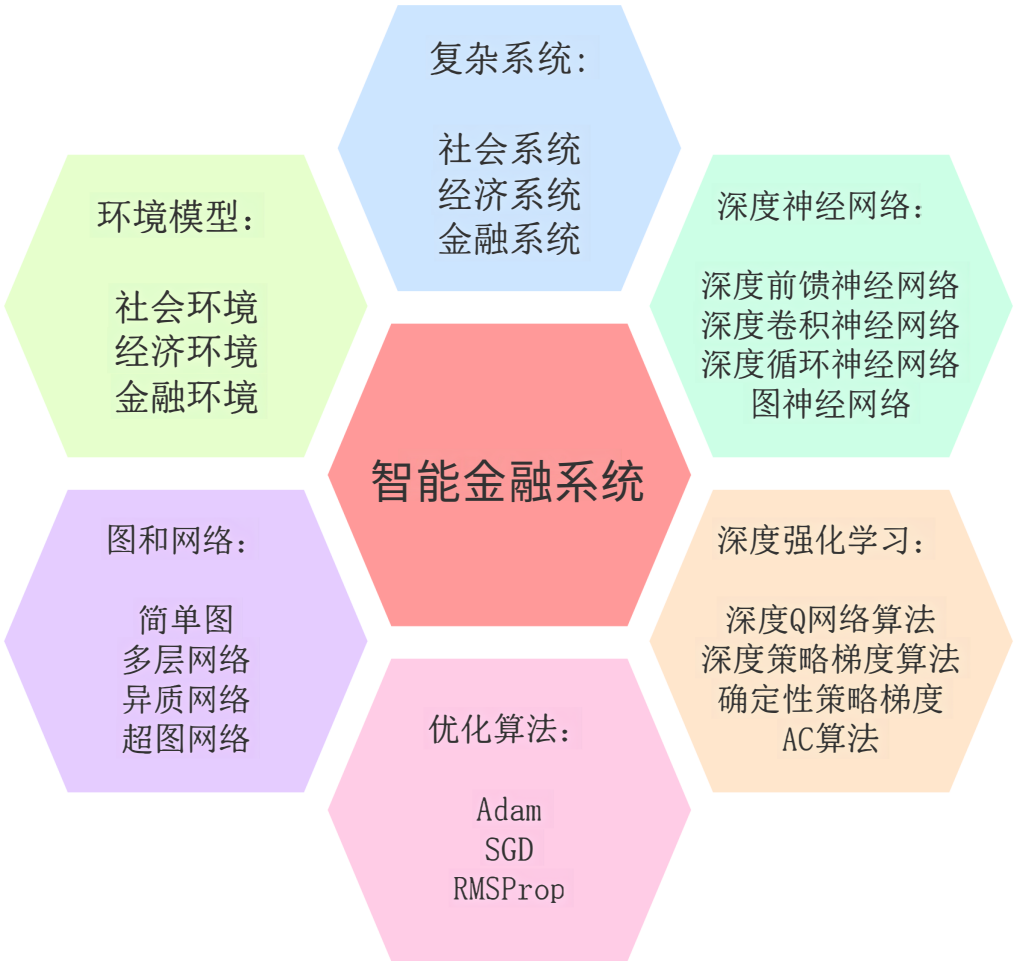


图 4: PPO2模型测试结果

模型改进


复杂系统：
社会系统
经济系统
金融系统
环境模型：
社会环境
经济环境
金融环境
深度神经网络：
深度前馈神经网络
深度卷积神经网络
深度循环神经网络
图神经网络
智能金融系统
图和网络：
简单图
多层网络
异质网络
超图网络
深度强化学习：
深度Q网络算法
深度策略梯度算法
确定性策略梯度
AC算法
优化算法：
Adam
SGD
RMSProp

掌握的问题

1. 确定性策略函数和随机性策略函数的区别是什么？
2. 简述随机性策略梯度定理。
3. 策略梯度优化算法中基线函数的作用是什么？
4. TRPO算法和PPO算法主要区别是什么？
5. PPO1和PPO2的主要区别是什么？
6. 研究深度神经网络结构对智能体投资收益的影响。
7. 运用策略梯度优化算法实现一个智能交易强化学习系统。