第8章
Actor-Critic 算法

周炜星 谢文杰

华东理工大学金融学系

2023年秋

纲要

纲要

Actor-Critic简介

深度强化学习算法门类庞杂，从简单到复杂、从离散空间到连续空间、从值函数到策略函数、从随机性策略到确定性策略，融合了众多深度学习模型和统计学习算法。

- Q学习算法（Q-learning）
- 深度Q神经网络算法（Deep Q Network，DQN）
- 策略梯度算法（Policy Gradient）
- 确定性策略梯度算法（Deterministic Policy Gradient）
- 深度确定性策略梯度算法（Deep Deterministic Policy Gradient）
- 孪生延迟深度确定性策略梯度算法（Twin Delayed DDPG）

Actor-Critic简介

- 在众多深度强化学习算法中，基于值函数的深度强化学习算法DQN具有重要历史地位；基于策略梯度的深度学习算法PPO和DDPG也得到了广泛应用。
- 深度强化学习算法的目标是学习智能体策略函数，智能体在与环境交互的过程中做出最优动作，获得最大累积收益。
- Actor-Critic（AC）算法，翻译成“行动者-评论家算法”或者“演员-评论家算法”。
- Actor-Critic算法框架结合了值函数和策略函数，迭代优化策略函数和值函数，是一类非常通用的强化学习算法框架，影响了很多深度强化学习算法，如DDPG等。

Actor-Critic简介

行动者（Actor）对应策略函数，产生行为动作；评论家（Critic）对应价值函数，评估动作的好坏或价值。状态-动作值函数表示为：

$$Q(s,a) = Q_w(s,a). \tag{1}$$

一般情况下，我们采用神经网络$Q_w(s,a)$表示，神经网络模型参数为w，$Q_w(s,a)$近似智能体在状态s下动作a的价值$Q(s,a)$，即期望累积收益。

确定性策略函数可以表示为：

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

智能体在给定状态下的确定性策略函数输出一个动作a。随机性策略函数表示为：

$$p(a|s) = \pi_\theta(s,a), \tag{3}$$

智能体在给定状态下的随机性策略函数输出动作a的概率分布。

纲要

AC算法介绍

蒙特卡罗方法可以估计值函数，需要采样完整路径（Trajectory or Episode）。AC方法采用了TD更新规则而避免了蒙特卡罗方法采样完整路径。我们估计状态-动作值函数，采用TD更新规则，计算TD目标：

$$Q_w^{\text{target}}(s,a) = r + \gamma Q_w(s',a'). \tag{4}$$

上式表明，智能体在状态s下执行动作a后转移至下一个状态s'，并获得即时奖励r，在状态s'下执行动作a'。
$Q_w(s,a)$和$Q_w^{\text{target}}(s,a)$都是估计值，我们将$Q_w^{\text{target}}(s,a)$作为目标来逼近。$Q_w^{\text{target}}(s,a)$包含了智能体一步的收益r，然后加上$\gamma Q_w(s',a')$，$Q_w(s',a')$也是动作值函数的估计值。但是相较于状态s和动作a的动作值估计$Q_w(s,a)$，估计值$r+\gamma Q_w(s',a')$在概率统计意义上比估计值$Q_w(s,a)$更加准确。

TD误差

经验数据中包含智能体行动信息和环境信息，基于经验数据优化策略函数能作出更好的预测和估计。如在DQN算法的改进算中，多步（n-step）DQN算法在实际应用中能够明显提升算法性能。我们用$Q_w(s,a)$去逼近目标值$Q_w^{\text{target}}(s,a)$，计算TD误差：

$$\delta = Q_w^{\text{target}}(s,a) - Q_w(s,a). \tag{5}$$

将式（4）代入上式可得：

$$\delta = r + \gamma Q_w(s',a') - Q_w(s,a). \tag{6}$$

为了使$Q_w(s,a)$尽可能逼近目标值$Q_w^{\text{target}}(s,a)$，需要通过状态-动作值函数$Q_w(s,a)$的参数更新使得TD误差越小越好。

状态-动作值函数参数更新

我们可以采用梯度下降算法更新状态-动作值函数$Q_w(s,a)$的参数：

$$w = w - \alpha_w \frac{1}{|\mathcal{B}|} \nabla \sum_{(s,a,r,s')\sim\mathcal{B}} \delta^2, \tag{7}$$

其中$\mathcal{B}$为经验数据集合。如果每次更新时采样一步经验数据(s,a,r,s')，则状态-动作值函数的更新规则为：

$$w = w + \alpha_w \delta \nabla Q_w(s,a). \tag{8}$$

公式中权重参数的更新分成了三部分：α_w、δ和$\nabla Q_w(s,a)$。

状态-动作值函数参数更新

- 在状态-动作值函数$Q_w(s,a)$的参数更新公式中，根据式(6)，如果$\delta > 0$，则$r + \gamma Q_w(s',a') - Q_w(s,a) > 0$，即$r + \gamma Q_w(s',a') > Q_w(s,a)$，说明$Q_w(s,a)$小于目标值，为了逼近目标值，$Q_w(s,a)$需要增大，因此$\delta \nabla Q_w(s,a)$为正的梯度方向，我们按照$w = w + \alpha_w \delta \nabla Q_w(s,a)$更新参数能够增加状态-动作值$Q_w(s,a)$。
- 反之，如果$\delta < 0$，则$r + \gamma Q_w(s',a') - Q_w(s,a) < 0$，即$r + \gamma Q_w(s',a') < Q_w(s,a)$，说明$Q_w(s,a)$大于目标值，$Q_w(s,a)$为了逼近目标值，需要减小$Q_w(s,a)$。因为$\delta \nabla Q_w(s,a)$为负的梯度方向，按照$w = w + \alpha_w \delta \nabla Q_w(s,a)$更新参数能够减小状态-动作值$Q_w(s,a)$。

策略函数参数更新

AC算法同时学习状态-动作值$Q_w(s,a)$和策略函数$\pi_\theta(s,a)$，上一节确定了值函数$Q_w(s,a)$的参数更新公式，我们接下来考虑策略函数$\pi_\theta(s,a)$参数θ的更新公式。在随机性策略梯度算法中，已对策略函数$\pi_\theta(s,a)$的参数θ更新进行了大量分析，基于梯度上升算法更新策略函数参数θ：

$$\theta = \theta + \alpha_\theta \delta \nabla \log \pi_\theta(s,a). \tag{9}$$

公式中参数θ的增量更新分成三个部分：α_θ、$\nabla \log \pi_\theta(s,a)$和$\delta$。超参数$\alpha_\theta$为智能体的策略函数学习率。学习率$\alpha_\theta$过大，学习曲线波动过大且不易收敛。学习率$\alpha_\theta$过小，过早陷入了局部最优解。

策略函数参数更新

- 策略函数$\pi_\theta(s,a)$的参数更新公式中，如果$\delta > 0$，则$r + \gamma Q_w(s',a') > Q_w(s,a)$，概率上更准确的目标值$r + \gamma Q_w(s',a')$大于当前估计值，此类动作$a$需要增加发生概率$\pi_\theta(s,a)$，因此$\delta \nabla \log \pi_\theta(s,a)$为正梯度方向，按照$\theta = \theta + \alpha_\theta \delta \nabla \log \pi_\theta(s,a)$更新能够增加动作$a$的发生概率$\pi_\theta(s,a)$。
- 策略函数$\pi_\theta(s,a)$的参数更新公式中，如果$\delta < 0$，则$r + \gamma Q_w(s',a') < Q_w(s,a)$，概率上更准确的目标值$r + \gamma Q_w(s',a')$小于当前估计值，此类动作$a$需要减少发生概率，因此$\delta \nabla \log \pi_\theta(s,a)$为负梯度方向，按照$\theta = \theta + \alpha_\theta \delta \nabla \log \pi_\theta(s,a)$更新能够减少动作$a$的发生概率$\pi_\theta(s,a)$。

AC算法伪代码

Algorithm 22: AC 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

深度神经网络模型 $Q_{\boldsymbol{w}}(s,a)$ 近似值函数，作为评论家

深度神经网络模型 $\pi_{\boldsymbol{\theta}}(s,a)$ 近似策略函数，作为行动者

Output: 最优策略 $\pi_{\boldsymbol{\theta}}$

初始化状态-动作值函数的参数 $\boldsymbol{w}$ 和策略函数的参数 $\boldsymbol{\theta}$

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

  % 智能体基于策略 $\pi_{\boldsymbol{\theta}}(s,a)$ 与环境交互，获得轨迹序列 (s,a,r,s')

  for $j = 0, 1, 2, \ldots$ **do**

    策略 $\pi_{\boldsymbol{\theta}}(s)$ 产生动作 a，获得即时奖励 r，以及下一个状态 s'

    策略 $\pi_{\boldsymbol{\theta}}(s')$ 产生动作 a'

    计算 TD 误差：$\delta = r + \gamma Q_{\boldsymbol{w}}(s',a') - Q_{\boldsymbol{w}}(s,a)$

    更新值函数模型参数：$\boldsymbol{w} = \boldsymbol{w} + \alpha_{\boldsymbol{w}} \delta \nabla Q_{\boldsymbol{w}}(s,a)$

    更新策略函数模型参数：$\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} \delta \nabla \log \pi_{\boldsymbol{\theta}}(s,a)$

    $a = a'$，$s = s'$

纲要

A2C（Advantage Actor-Critic）算法分析

A2C全称为优势行动者-评论家算法（Advantage Actor-Critic，A2C），是为了使智能体更高效地学习到最优化策略函数，而在AC算法的基础上进行了改进。A2C算法中智能体更新策略函数的目标函数为：

$$J(\theta) = \sum_s \mu(s) \sum_a Q(s,a)\pi_\theta(s,a). \tag{10}$$

其中，$\mu(s)$为状态s的稳定概率分布，$Q(s,a)$为状态-动作值函数。目标函数的梯度为：

$$\nabla_\theta J(\theta) = \sum_s \mu(s) \sum_a Q(s,a)\nabla\pi_\theta(s,a). \tag{11}$$

其中，$\nabla\pi_\theta(s,a)$为随机性策略函数$\pi_\theta(s,a)$的梯度。

A2C（Advantage Actor-Critic）算法分析

目标函数的梯度为：

$$\nabla_\theta J(\theta) = \sum_s \mu(s) \sum_a Q(s,a) \nabla \pi_\theta(s,a). \tag{12}$$

其中，$\nabla \pi_\theta(s,a)$为随机性策略函数$\pi_\theta(s,a)$的梯度。策略函数$\pi_\theta(s,a)$的参数θ按照梯度方向更新，能够增加状态s下动作a的概率值$\pi_\theta(s,a)$。策略函数的参数更新可以写作：

$$\theta = \theta + \alpha_\theta \nabla_\theta J(\theta). \tag{13}$$

A2C（Advantage Actor-Critic）算法分析

A2C算法在REINFORCE等算法的基础上引入基线函数，构建了优势函数，改进后策略梯度公式如下所示：

$$\begin{aligned}\nabla_\theta J(\theta) &= \sum_s \mu(s) \sum_a Q(s,a) \nabla \pi_\theta(s,a) \\ &= \sum_s \mu(s) \sum_a (Q(s,a) - b(s)) \nabla \pi_\theta(s,a).\end{aligned} \tag{14}$$

在上式中，状态-动作函数值$Q(s,a)$减去基线函数$b(s)$不影响目标函数梯度值，可简单证明如下。

A2C（Advantage Actor-Critic）算法分析

目标函数梯度公式引入基线函数后的新增部分为：

$$\begin{aligned}\sum_s \mu(s) \sum_a b(s) \nabla \pi_\theta(s,a) &= \sum_s \mu(s) b(s) \sum_a \nabla \pi_\theta(s,a) \\ &= \sum_s \mu(s) b(s) \nabla \sum_a \pi_\theta(s,a) \\ &= \sum_s \mu(s) b(s) \nabla 1 = 0.\end{aligned} \tag{15}$$

A2C（Advantage Actor-Critic）算法分析

在推导过程中，用到了概率分布函数的归一化条件，即在给定状态s时，所有动作发生的概率之和为1：

$$\sum_{a} \pi_\theta(s, a) = 1, \tag{16}$$

公式（14）中的基线函数不包含策略函数参数θ，因此基线函数$b(s)$对目标函数梯度没有影响。最简单的基线函数为一个常数：

$$b(s) = b_0. \tag{17}$$

常数b_0可通过智能体采样经验轨迹进行估计。

优势函数和基线函数

在程序实现过程中，基线函数的常数值b_0需要人为设定或通过智能体采样经验轨迹进行估计，这增加了算法稳定收敛的难度，因为人为制定的常数很有可能不能增加算法稳定性。一般来说，我们可以设定基线函数为状态值函数：

$$b(s) = V(s). \tag{18}$$

状态值函数$V(s)$不包含策略函数的参数θ，因此不影响目标函数梯度。因此，我们需要计算或估计状态值函数$V(s)$。选择状态值函数$V(s)$作为基线函数时，公式

$$V(s) = \sum_{a \in \mathcal{A}} \pi_\theta(s,a) Q_{\pi_\theta}(s,a) \tag{19}$$

中的状态值函数$V(s)$是状态-动作值函数$Q(s,a)$在动作空间上的期望值，或平均值。

优势函数和基线函数

状态-动作值函数$Q(s,a)$减去$V(s)$后，可以得到优势函数（Advantage Function）：

$$A(s,a) = Q(s,a) - V(s), \tag{20}$$

优势函数$A(s,a)$可以看作是动作a的动作价值$Q(s,a)$高于所有动作平均价值$V(s)$的优势程度。我们对于高于平均值的动作增加其选择概率，对于小于平均值的动作则减小其选择概率。所以，目标函数梯度可以表示为：

$$\begin{aligned} \nabla_\theta J(\theta) &= E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)(Q(s,a) - V(s))] \\ &= E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) A(s,a)]. \end{aligned} \tag{21}$$

A2C（Advantage Actor-Critic）算法分析

在A2C算法实现过程中，需要考虑如何实现对优势函数值的估计或者近似计算，类似于策略梯度算法的值函数在更新过程中使用了TD误差，A2C算法中同样采用TD误差进行模型参数更新：

$$\delta_{\pi_\theta} = r + \gamma V_{\pi_\theta}(s') - V_{\pi_\theta}(s), \tag{22}$$

公式中状态值函数$V_{\pi_\theta}(s)$的下标表示基于策略函数π_θ采样经验轨迹数据进行状态值函数估计。分析发现，TD误差计算与优势函数之间存在关联关系：

$$\begin{aligned} E_{\pi_\theta}[\delta_{\pi_\theta}|s,a] &= E_{\pi_\theta}[r + \gamma V_{\pi_\theta}(s')|s,a] - V_{\pi_\theta}(s) \\ &= Q_{\pi_\theta}(s,a) - V_{\pi_\theta}(s) \\ &= A_{\pi_\theta}(s,a). \end{aligned} \tag{23}$$

A2C（Advantage Actor-Critic）算法分析

在A2C算法实现过程中，我们用深度神经网络模型参数化状态值函数$V_w(s)$，其中w为深度神经网络模型参数。因此，优势函数可以重新写为：

$$A(s,a) = \delta = r + \gamma V_w(s') - V_w(s). \tag{24}$$

策略函数π_θ的参数更新可以写为:

$$\begin{aligned} \theta &= \theta + \alpha_\theta \nabla_\theta J(\theta) \\ &= \theta + \alpha_\theta E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\delta]. \end{aligned} \tag{25}$$

状态值函数$V_w(s)$的参数更新可以表示为：

$$w = w + \alpha_w \delta \nabla V_w(s). \tag{26}$$

A2C算法伪代码

Algorithm 23: A2C 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
深度神经网络 $V_{\boldsymbol{w}}(s)$ 近似状态值函数，作为评论家
深度神经网络 $\pi_{\boldsymbol{\theta}}(s,a)$ 近似策略函数，作为行动者
Output: 最优策略 $\pi_{\boldsymbol{\theta}}$

初始化状态值函数 $V_{\boldsymbol{w}}(s)$ 的参数 $\boldsymbol{w}$ 和策略函数 $\pi_{\boldsymbol{\theta}}(s)$ 的参数 $\boldsymbol{\theta}$
for $k=0,1,2,3,\cdots$ **do**
  % 智能体基于策略 $\pi_{\boldsymbol{\theta}}(s)$ 与环境交互，获得轨迹序列 (s,a,r,s')
  for 轨迹中每一步 **do**
    策略 $\pi_{\boldsymbol{\theta}}(s)$ 产生动作 a，获得及时奖励 r，以及下一个状态 s'
    计算 TD 误差：$\delta = r+\gamma V_{\boldsymbol{w}}(s') - V_{\boldsymbol{w}}(s)$
    更新状态值函数网络：$\boldsymbol{w} = \boldsymbol{w} + \alpha_{\boldsymbol{w}}\delta\boldsymbol{\nabla} V_{\boldsymbol{w}}(s)$
    更新策略网络：$\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}}\delta\boldsymbol{\nabla}\log\pi_{\boldsymbol{\theta}}(s,a)$
    $s = s'$

纲要

A3C介绍

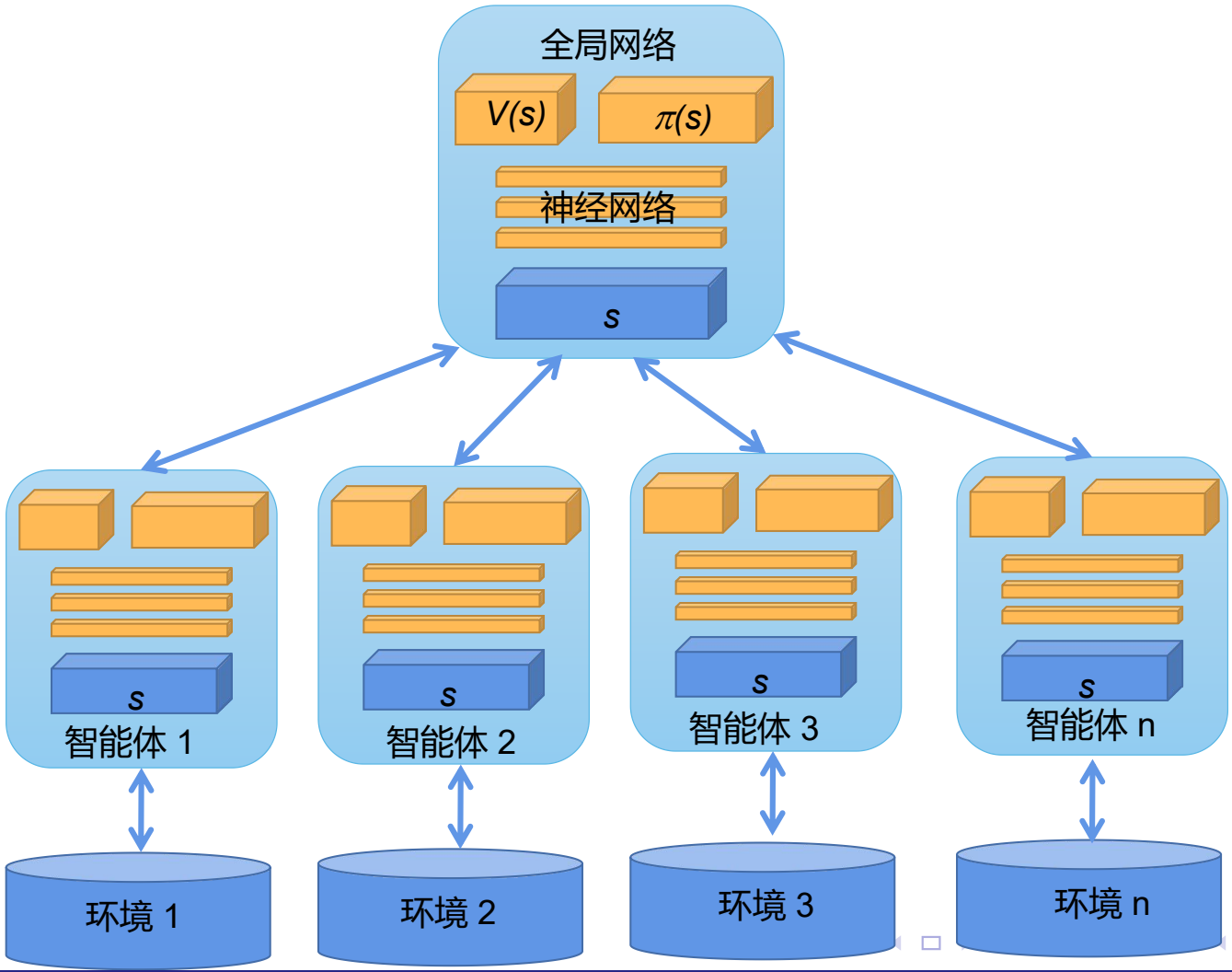
- 在AC框架下，智能体同时学习值函数和策略函数，值函数是指AC算法中的状态-动作值函数或A2C算法中的状态值函数。
- 值函数辅助智能体更新策略函数的参数，智能体依靠策略函数与环境交互获得更好的经验轨迹样本数据来更新值函数和策略函数参数。
- 值函数和策略函数共同协作，共同进步，取长补短，互相成就。
- 在深度强化学习算法DQN、DPG、AC、A2C算法的演化过程中，算法在原理和模型结构上都作了大量改进和优化。
- 多个智能体与环境进行交互具有较多优势。智能体在学习过程中，与环境交互获得采样轨迹样本是学习的关键，多个智能体同时采样本身就能够加快学习效率，也是并行或者分布式计算最直接的优势。

A3C算法的改进和优化

A3C算法全称是异步优势动作评价算法（Asynchronous Advantage Actor-Critic）。相较于A2C（Advantage Actor-Critic）算法，A3C算法多了关键字Asynchronous，因此异步更新是A3C算法的核心和需要重点理解的技术。

- 智能体分成了两类，一类是中心主智能体（Global Network），另一类是子智能体（Local Network）。
- 子智能体与各自的环境交互获得采样的经验轨迹样本数据，主智能体负责和子智能体交互，异步更新主智能体决策函数参数和值函数参数。
- 子智能体与各自的环境交互获得轨迹数据，计算值函数和策略函数神经网络参数的梯度，但是梯度信息不用来更新子智能体的深度神经网络。
- 子智能体独立地使用累积的梯度分别更新主智能体的深度神经网络模型参数。

异步更新优化



网络模型结构优化

- 在框架图1中，A3C算法对深度神经网络模型结构也进行了优化。
- 主智能体和子智能体的决策函数和值函数对应的神经网络有部分网络架构是一样的，即策略函数和值函数共享部分参数。
- 主智能体和子智能体的深度神经网络结构完全一样，可以互相传递函数梯度信息进行参数更新。
- 图1 中的$V(s)$表示状态值函数，采用带有参数w的深度神经网络模型$V_w(s)$近似。
- 策略函数用$\pi_\theta(s)$表示，采用带有参数θ的深度神经网络模型近似。
- 策略网络和值函数网络参数和梯度更新细节与A2C算法类似。

网络模型参数更新

A3C算法采用TD参数更新规则更新状态值函数$V_w(s)$参数，TD误差计算公式如下：

$$\delta_{\pi_\theta} = r + \gamma V_{\pi_\theta}(s') - V_{\pi_\theta}(s), \tag{27}$$

其中，r为即时奖励，s'为下一个状态。然后，通过TD误差进行状态值函数参数w的更新：

$$w = w + \alpha_w \delta \nabla V_w(s). \tag{28}$$

公式中$\nabla V_w(s)$为状态值函数梯度，按照梯度方向更新参数，可使得状态值$V_w(s)$不断增大。

网络模型参数更新

A3C算法与AC算法、A2C算法类似，策略函数参数θ更新公式如下：

$$\theta = \theta + \alpha_\theta \nabla \log \pi_\theta(s, a)\delta. \tag{29}$$

策略函数参数θ具有相似的更新规则，但是A3C采用异步更新，子智能体共享梯度信息给主智能体更新深度神经网络模型参数。

- 在状态值函数参数更新公式中，如果$\delta > 0$，那么$r + \gamma V_w(s') - V_w(s) > 0$，即$r + \gamma V_w(s') > V_w(s)$，说明$V_w(s)$小于目标值，$V_w(s)$为了逼近TD目标值，需要增大，因此$\delta \nabla V_w(s)$为正梯度方向，按照$w + \alpha_w \delta \nabla V_w(s)$更新参数$w$能够增加状态值$V_w(s)$并逼近TD目标值。

A3C算法伪代码

Algorithm 24: A3C 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
子智能体状态值函数 $V_{\boldsymbol{w}}(s)$ 和策略函数 $\pi_{\boldsymbol{\theta}}(s,a)$
主智能体状态值函数 $V_{\boldsymbol{w}'}(s)$ 和策略函数 $\pi_{\boldsymbol{\theta}'}(s,a)$
Output: 最优策略 $\pi_{\boldsymbol{\theta}'}$

初始化子智能体状态值函数 $V_{\boldsymbol{w}}(s)$ 的参数 $\boldsymbol{w}$ 和策略函数 $\pi_{\boldsymbol{\theta}}(s,a)$ 的参数 $\boldsymbol{\theta}$，以及主智能体网络的参数 $\boldsymbol{\theta}'$ 和 $\boldsymbol{w}'$
设定全局更新次数 $T=0$
子智能体步数 $t=1$
for $k=0,1,2,\cdots,T_{\max}$ **do**
  重新设定累积梯度 $\mathrm{d}\boldsymbol{\theta}=0$ 和 $\mathrm{d}\boldsymbol{w}=0$
  更新子智能体网络参数 $\boldsymbol{\theta}=\boldsymbol{\theta}'$ 和 $\boldsymbol{w}=\boldsymbol{w}'$
  子智能体线程时间步为 $t_{\text{start}}=t$
  子智能体获得初始状态 s_t
  for $j=0,1,2,\cdots,t_{\max}$ **do**
    子智能体基于策略 $\pi_{\boldsymbol{\theta}}(s,a)$ 与环境交互，获得经验数据序列 (s,a,r,s')
    策略 $\pi_{\boldsymbol{\theta}}(s)$ 产生动作 a，获得及时奖励 r，以及下一个状态 s'
    $t\leftarrow t+1$
    $T\leftarrow T+1$
    计算 TD 误差：$\delta=r+\gamma V_{\boldsymbol{w}}(s')-V_{\boldsymbol{w}}(s)$
    累计值函数网络梯度：$\mathrm{d}\boldsymbol{w}=\mathrm{d}\boldsymbol{w}+\delta\nabla V_{\boldsymbol{w}}(s)$
    累计策略网络梯度：$\mathrm{d}\boldsymbol{\theta}=\mathrm{d}\boldsymbol{\theta}+\delta\nabla\log\pi_{\boldsymbol{\theta}}(s,a)$
    if 需要进行异步更新 **then**
      $\boldsymbol{w}'=\boldsymbol{w}'+\alpha_{\boldsymbol{w}}\mathrm{d}\boldsymbol{w}$
      $\boldsymbol{\theta}'=\boldsymbol{\theta}'+\alpha_{\boldsymbol{\theta}}\mathrm{d}\boldsymbol{\theta}$

纲要

SAC算法伪代码

Algorithm 25: SAC 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-动作值函数 $Q_{w_1}(s,a)$ 的参数 w_1 和 $Q_{w_2}(s,a)$ 的参数 w_2

初始化策略函数 $\pi_\theta(s)$ 的参数 θ

初始化策略目标网络和动作值目标网络：$w_1^- = w_1$、$w_2^- = w_2$ 和 $\theta^- = \theta$

Output: 最优策略函数 $\pi_\theta(s,a)$

清空经验池 $\mathcal{D}$

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

  智能体基于策略 π_θ，获得动作：

$$a = \text{clip}\left(\pi_\theta(s) + \text{clip}(\epsilon, -c, c), a_{\text{Low}}, a_{\text{High}}\right) \tag{8.51}$$

  智能体与环境交互，获得轨迹集合 $\mathcal{D}_k = \{\tau_i\}$

  构建经验池 $\mathcal{D}$，其中经验数据用五元组 (s,a,r,s',d) 表示。d 作为一个确定状态 s' 是否为轨迹终点的标志：当 s' 为终点时，$d=1$，否则 $d=0$

  if 需要更新参数 **then**

    for $j = 0, 1, 2, \ldots$ **do**

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

      计算 TD 目标值：

$$y(r,s',d) = r(s,a) + \gamma(1-d)\left(\min_{i=1,2} Q_{w_{i,-}}(s',\tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s')\right) \tag{8.52}$$

      采样梯度下降算法更新动作值参数：

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

$$w_i = w_i - \alpha_w \nabla \frac{1}{B} \sum_{(s,a,r,s',d)\sim\mathcal{B}} \left[\left(Q_{w_i}(s,a) - y(r,s',d)\right)^2\right] \tag{8.53}$$

      更新目标函数的参数，采样软更新方法：

$$\begin{aligned} w_1^- &= \rho w_1^- + (1-\rho) w_1 \\ w_2^- &= \rho w_2^- + (1-\rho) w_2 \\ \theta^- &= \rho \theta^- + (1-\rho)\theta \end{aligned} \tag{8.54}$$

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

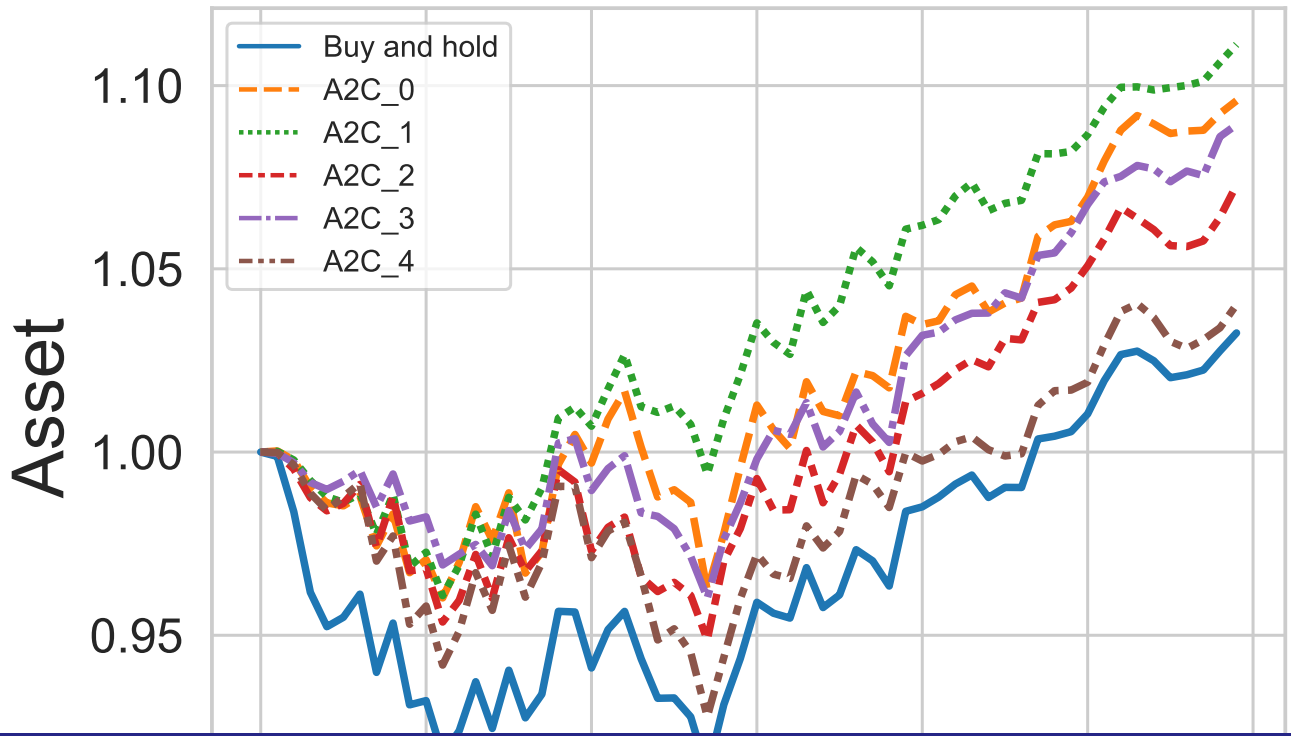
        采用梯度上升更新策略函数参数：

$$\theta = \theta + \alpha_\theta \nabla \sum_{(s,a,r,s',d)\sim\mathcal{B}} \left[Q_{w^-}(s, \pi_{\theta^-}(s)) - \alpha \log \pi_\theta(\tilde{a}'_\theta(s)|s')\right] \tag{8.55}$$

纲要

模型测试

图2给出了在相同参数下独立训练的5个A2C智能体的投资策略在测试数据中的资产价值变化情况。图中横坐标为时间步数，纵坐标为单个投资周期内的资产价值。同样，为了保证模型测试的有效性，将训练集和测试集进行了严格划分。



课后习题

1. Actor-Critic与DQN的区别是什么？
2. AC、A2C和A3C的区别是什么？
3. SAC算法有哪些优点？
4. SAC算法与TD3算法异同是什么？
5. 深度强化学习算法采用什么策略获取海量、高质量的经验轨迹样本数据？
6. 运用AC、A2C或A3C算法实现一个智能交易强化学习系统。