中南大學
CENTRAL SOUTH UNIVERSITY

毕业设计(论文)

GRADUATION DESIGN(THESIS)

题　　目	基于深度强化学习的量化交易策略研究
学生姓名	黄辉晴
指导教师	邓磊、倪炜
学　　院	计算机学院
专业班级	软件工程 1705 班

本科生院制

二〇二一年六月

基于深度强化学习的量化交易策略研究

摘要

计算机科学和人工智能的发展，对各行各业产生了深远的影响，金融交易领域也不例外。西蒙斯创办的文艺复兴基金便是计算机在金融交易领域应用的最佳印证。然而，在一个复杂且多变的金融市场环境下，构建一个可以持续盈利的交易策略是非常具有挑战性的。我们构建的策略很有可能短期表现良好，但长期来看却是亏损的。

基于上述挑战，本文试图使用深度强化学习算法，寻找到切实可行的量化交易策略。交易时所用到的股票为流动性充足的上证 50 成分股，使用 Tushare API 进行数据的调用。所使用到的强化学习算法包括： PPO、 A2C、SAC、TD3 和 DDPG。在验证阶段，我们会将强化学习算法所得到的交易策略和上证 50 指数进行比较，并以 Sharpe 比率、最大回撤、累计收益率和 Omega 比率等指标作为评判标准。结果表明，强化学习交易策略显著优于上证 50 指数的表现。其中，PPO 智能体的 Sharpe 比率和 Omega 比率最高，比其他智能体更加适用于金融交易领域。

关键词：强化学习　深度强化学习　量化交易　量化投资

Quantitative trading strategies based on deep reinforcement learning

Abstract

The development of computer science and artificial intelligence has had a profound impact on all walks of life, and the field of financial transactions is no exception. Simmons's Renaissance Fund is a prime example of how computers can be used in financial transactions. However, in a complex and volatile financial market environment, it's a huge challenging to construct a trading strategy that can be consistently profitable. We build a strategy that is likely to perform well in the short term but lose money in the long term.

Based on the challenges above, this paper attempts to use deep reinforcement learning algorithm to find a feasible quantitative trading strategy. The stocks used in the transaction are the 50 constituent stocks of Shanghai Stock Exchange with sufficient liquidity, and the Tushare API is used for pulling data. The reinforcement learning algorithms used include PPO, A2C, SAC, TD3 and DDPG. In the validation stage, we will compare the trading strategies obtained by reinforcement learning algorithm with the SSE 50 Index, and take Sharpe ratio, maximum retracement, cumulative return rate and Omega ratio as the evaluation criteria. The results show that the reinforcement learning trading strategy significantly outperforms the SSE 50 index. Among them, PPO agents have the highest Sharpe ratio and Omega ratio, which is more applicable to the financial transaction field than other agents.

Key words: reinforcement learning　deep reinforcement learning　quantitative trading　quantitative investment

目录

第 1 章　绪论

1.1　研究工作的背景与意义

构建一个能长期盈利的股票自动交易策略对投资公司和对冲基金来说是至关重要的。交易策略可以被用于优化资本配置和最大化投资收益，如预期收益。同时也可以被用于风险的评估，从而降低投资的风险。然而，对于分析师来说，在一个复杂而动态的股票市场中考虑所有相关因素是具有挑战性的。

当前阶段，现有的相关工作并不令人满意。[1]中描述了一种传统方法，其实现步骤分为两步。首先，计算股票预期收益和股票价格的协方差矩阵。然后，在给定的风险比下最大化收益，或在给定的收益下最小化风险，得到最佳的投资组合配置策略。然而，这种方法的实现是复杂且消耗较高的，因为基金经理可能希望在每个时间点都对决策进行修改，并考虑其他因素，如交易成本。另一种股票交易的方式是将其建模为马尔科夫决策过程(MDP)，并利用动态规划推导出最优策略[2]。 然而，该模型的可扩展性受到限制，因为股票市场具有较大的状态空间。

近年来，机器学习和深度学习算法被广泛应用于构建金融市场的预测和分类模型。基本面数据(收益报告)和其他数据(市场新闻、学术图表数据、信用卡交易和 GPS 流量等)与机器学习算法相结合，以提取新的投资阿尔法值或预测公司的未来表现[3]。因此，生成一个可用来做预测的阿尔法因子来挑选相应的股票。然而，这些方法只专注于挑选高绩效股票，而不是在选定的股票之间配置交易头寸或股票。换句话说，经过训练的机器学习模型并不能直接用来对资金进行交易。

量化交易的本质是分析金融市场、了解市场运动规律来辅助投资决策。目前，强化学习在投资决策领域的应用较少，在金融投资领域缺少独立的、系统的理论分析框架，处于探索、创建阶段。本文基于深度强化学习对量化交易进行研究，为量化投资的策略构建以及深度强化学习在金融投资领域的应用进行验证，有助于强化学习理论在金融领域的扩展，拓宽了强化学习的理论应用，同时丰富了金融投资的工具，具有很强的理论和应用研究价值。

1.2 国内外研究历史与现状

Varela （2016）基于 Q-learning 构建投资策略，策略在投资环境中根据反馈进行学习，但当状态空间很大时会呈现出一些局限性[4]。 Igor Halperin（2017）结合了强化学习的 Q-learning 方法和 Black-Scholes-Merton 模型的思想，将期权定价和对冲转化为股票和现金的动态复制组合的最优再平衡问题，提出了基于动态规划和强化学习的离散时间期权套期保值定价模型，该模型学习为期权复制投资组合动态优化风险，并根据调整后的收益对期权进行定价。后续又将逆强化学习应用到模型中，通过观察价格和交易者采取的动作来进一步优化模型。量化投资传入国内时机器学习正蓬勃发展，不少国内学者在量化投资研究中运用机器学习算法[5]。 岳登（2017）等学者将深度强化学习运用到金融信号的表示和学习中，使用深度学习感知金融市场的动态变化并完成特征提取，使用强化学习进行决策，在股票市场和商品期货市场上验证了该方法的鲁棒性。

最近，深度强化学习在金融市场中的应用考虑的是离散或连续的状态和动作空间，并且采用的算法为 critic-only 算法、actor-only 算法或 actor-critic 算法[26]中的一种。连续动作空间的学习模型和离散动作空间的学习模型相比，前者可以输出细粒度更高的动作。

较为常见的，是 critic-only 学习算法，解决的是离散行动空间下的问题，例如，Deep Q-learning (DQN)及其改进，如：在某股票或某资产[6]下训练一个智能体。critic-only 算法的思想是使用 Q-value 函数来学习最优的行动选择策略，在给定的状态下，使预期未来回报最大化。DQN 不是计算一个 state-action value 表，而是使用神经网络函数，逐步减少 Q-value 和 target Q-value 之间的差值。这种算法的局限性在于它只适用于离散的、有限的状态空间和动作空间，这显然不适用于交易一个大型的股票投资组合，因为价格的变化是连续的。

在[7]中使用的是 actor-only 的算法，想法是智能体直接学习最优策略本身。神经网络学习的不是 Q 值，而是策略。策略是一种概率分布，本质上来说，是给定状态的策略，即采取所有行动的可能性。同时，我们也可以使用循环强化学习，打破维度的限制，提高交易效率。 actor-only 算法可以处理连续的动作空间问题。

actor-critic 算法最近被广泛应用于金融领域[8][28][29][30]。算法的核心思想是同时更新代表策略的 actor 网络和代表价值函数的 critic 网络。critic 对价值函数进行估计，而 actor 用策略梯度更新在 critic 指导下的策略概率分布。随着时间的推移，actor 学会了采取更好的行动，而 critic 也能更好地评价这些行动。事实证明，actor-critic 的方法能够学习和

适应高维和复杂的环境，并已被用于玩流行的电子游戏，如 Doom [9]。 因此，actor-critic 算法在大规模股票投资组合交易中有很大的应用前景。

1.3 本文的研究内容

在使用强化学习构建股票交易模型的过程中，我们首先需要将股票交易过程建模为马尔科夫决策过程(MDP)。交易目标暂且定义为在回撤较小的情况下求期望收益的最大值。

1.3.1 股票交易的 MDP 模型

以股票市场的动态随机性进行建模，我们采用如下的马尔科夫决策过程（MDP）：

- State $s = [p, h, b]$：包含股票 $p \in R_+^D$ 的价格信息的集合，股票持有量 $h \in Z_+^D$ ，余额$b \in R_+$，其中 D 为我们在市场中的持仓量，Z^+表示非负整数。
- Action a：对 D 中所有股票进行操作的集合。每只股票的动作空间为卖出、买入和持有，分别导致持有量 h 减少、增加和不变。
- Reward $\mathrm{r(s, a, s')}$： 在状态 s 处执行动作 a 并到达新状态 s' 时，投资组合将发生变化。该投资组合的总额由所持有的股票市值之和 $p^T h$ 和余额 b 两部分构成。
- Policy $\pi(\mathrm{s})$：股票在状态 s 下的交易策略。它本质上是 a 在状态 s 下的概率分布。
- Q 值 $Q_\pi(\mathrm{s,a})$：在状态 s 采取策略 π 所得到的动作 a 的预期回报。
- 股票交易过程的状态转换如图 1-1 所示。在每个状态下，对投资组合中的股票 d $(d = 1, \dots, D)$ 采取三种可能的动作之一。

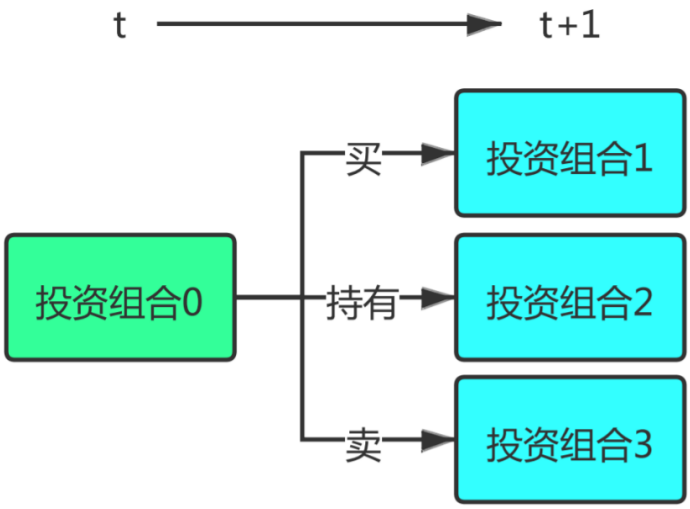

图 1-1　t 时刻执行不同的操作，在 t+1 时刻会产生三种不同的投资组合

- Selling ：在当前持有的股份数中，出售k（$k \in [1, h[d]]$，其中 $d = 1, \ldots,$ D)份，其中 k 必须是一个整数。导致 $h_{t+1} = h_t - k$。
- Holding ： $k = 0$， $h_{t+1}[d] = h_t[d]$。
- Buying ： 买入 k 股，导致 $h_{t+1}[d] = h_t[d] + k$

在 t 时刻采取行动，根据执行的 $action$ 和股票价格的更新，在 $t + 1$ 时刻组合值从“投资组合 0”变为“投资组合 1”、“投资组合 2”或“投资组合 3”。

1.3.2　纳入股票交易的约束条件

以下假设使用约束用来反映股票市场的实际情况，如：市场流动性、交易成本等。

- 市场流动性：交易可以以收盘价迅速执行。我们假定股票市场不会受到我们强化交易代理的影响。
- 余额非负 $b \geq 0$：可执行的动作不应导致余额成为负数。在 t 时刻，交易股票时可选的动作分别为卖出 S 、买入 B 、持有 H 三种，其中 $S \cup B \cup H = \{1, \ldots, D\}$，互不重叠。设 $p_t^B = [p_t^i : i \in B]$ 和 $k_t^B = [k_t^i : i \in B]$分别为当前要买的股票的价格和数量，与此同时，我们也可以定义 p_t^S 和 k_t^S 为当前要卖的股票的价格和数量， p_t^H 和 k_t^H 为当前持有的股票的价格和数量。因此，余额非负的约束可以表示为：

$$b_{t+1} = b_t + (p_t^S)^T k_t^S - (p_t^B)^T k_t^B \geq 0 \tag{1-1}$$

- 交易手续费：每笔交易都会产生手续费[32]。手续费分为多个类别，如交易佣金、印花税等。我们假设我们的交易成本为每笔交易(买卖)金额的 0.1%，如[10]所示：

$$c_t = p^T\, k_t \times 0.1\% \tag{1-2}$$

1.3.3 将收益最大化作为交易目标

我们将奖励函数定义为在状态 s 执行动作 a 到达新状态 s' 时，投资收益的变化。目标是设计一种交易策略，使投资收益最大化：

$$r(s_t, a_t, s_{t+1}) = (b_{t+1} + p_{t+1}^T h_{t+1}) - (b_t + p_t^T h_t) - c_t \tag{1-3}$$

其中第一项和第二项分别表示 t+1 和 t 时刻投资组合的总价值。为了对收益进行进一步的分解，我们对 h_t 有如下定义：

$$h_{t+1} = h_t - k_t^S + k_t^B \tag{1-4}$$

根据式(1-1)中余额 b 的定义，那么式(1-3)可以重写为：

$$r(s_t, a_t, s_{t+1}) = r_H - r_S + r_B - c_t \tag{1-5}$$

其中

$$r_H = (p_{t+1}^H - p_t^H)^T h_t^H \tag{1-6}$$

$$r_S = (p_{t+1}^S - p_t^S)^T h_t^S \tag{1-7}$$

$$r_B = (p_{t+1}^B - p_t^B)^T h_t^B \tag{1-8}$$

从 t 时刻到 $t+1$ 时刻，$r_H, r_S\, r_B$ 分别表示持有、卖出、买入股票时组合价值的变化。由式(1-5)可得，我们需要购买并持有下一时刻价格会上涨的股票来增加投资组合的收益，卖出下一时刻价格会下跌的股票来降低投资组合的损失。

1.3.4 使用强化学习算法进行训练

算法的初始化如下所示。设 p_0 为 0 时刻股票价格的集合，b_0 为初始资金量。h 和 $Q(s,a)$ 都初始化为 0，在任何状态下的所有动作 $\pi(s)$ 为均匀分布。之后，$Q_\pi(s_t, a_t)$ 通过与外部环境的互动来进行学习。最优策略由 Bellman 方程给出，根据 Bellman 方程，执行动作 $\mathrm{a_t}$ 的期望奖励值是由奖励的期望 $r(s_t, a_t, s_{t+1})$，加上下一个状态的预期奖励 s_{t+1} 所得到的。并使用 $\gamma(0 < \gamma < 1)$ 做为折现因子，我们有如下方程：

$$Q_\pi(s_t, a_t) = E_{s_{t+1}}\left[r(s_t, a_t, s_{t+1}) + \gamma E_{a_{t+1}\sim\pi(s_{t+1})}\left[Q_\pi(s_{t+1}, a_{t+1})\right]\right] \tag{1-9}$$

我们的目标是挑选并使用深度强化学习算法，来构建交易策略，使投资组合的收益 $r(s_t, a_t, s_{t+1})$ 最大化。

1.4 本论文的结构安排

接下来的章节安排如下：第 2 章将会介绍强化学习和量化交易的一些基础知识。在

第3章，我们会详细说明基于强化学习的量化交易系统的构建，构建的步骤大体上包括：构建股票交易环境和实现交易算法。在第4章我们将会从多个方面对交易策略的结果进行评估。在第5章我们将会对当前的工作做出总结，并探讨之后所要做的工作。

第 2 章 强化学习和量化交易

2.1 强化学习

强化学习(Reinforcement Learning)，是一种和环境(environment)进行交互的学习方式，是在没有明确指导的情况下，学会如何通过经验(experience)去执行相应的动作(action）以获得最大奖励的问题。强化学习智能体（agent）需要同它所处的环境(environment)交互，并从中学习如何将累积的奖励（reward）最大化。

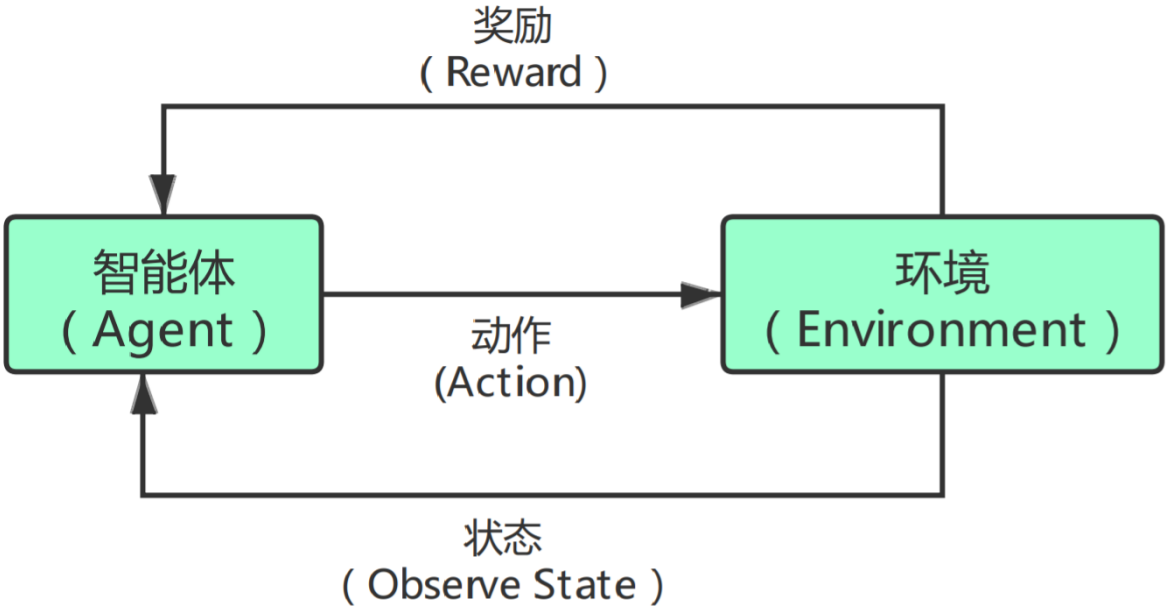

图 2-1 强化学习的学习过程

于此同时，由于 Deep Q-Network 的兴起，近些年来强化学习方面的研究变得越来越热。人工智能的其他领域也正尝试通过借鉴与利用强化学习中的去达到更好的效果。例如，AlphaGo 通过使用强化学习在围棋领域超过了人类的水平[11]，强化学习的概念也得以应用在生成对抗网络（Generative Adversarial Networks，GAN）的训练过程中[27]。

2.1.1 马尔科夫性质

在介绍马尔科夫性质之前，我们先对一些关键性的变量进行定义：令智能体的状态空间和动作空间分别表示为 S 和 A，那么，

模型(Model)：模型是智能体所处的环境的状态转移与奖励的数学模型，包括状态转移概率 $P(s'|s,a)$，表示状态 $s \in S$ 在执行动作 $a \in A$ 后，状态转移到 $s' \in S$ 后的概

率，以及在状态 $s \in S$ 执行动作 $a \in A$ 后得到的奖励 $R(s,a)$（确定的或随机的）。

策略(Policy)：策略是一个将智能体的状态映射到动作的函数 $\pi: S \to A$。

值函数(Value-function)：对于特定策略 π 和状态 $s \in S$ 的值函数 V_π 是智能体从状态 s 开始，遵循策略 π 所能获得的未来（衰减）奖励的累加。

而马尔科夫性质则可定义为：考虑一个遵循某个转移规律的随机过程 $(s_0, s_1, s_2, \ldots)$，当且仅当对于 $\forall_i = 1,2,\ldots$，$P(s_i|s_0,\ldots,s_{i-1}) = P(s_i|s_{i-1})$，我们才能称这个随机过程具有马尔科夫性质，即以包括当前状态在内的历史状态为条件，转移到下一状态的概率，与仅以当前状态为条件的转移到下一状态的概率相等。在这种情况下，当前状态是随机过程的历史的一个分布统计，也就是说“未来与过去无关，只与当前状态有关”。

2.1.2 马尔科夫过程（MP）

一般的情况下，马尔科夫过程是一个满足马尔科夫性质的随机过程，因此我们称马尔科夫过程是“无记忆的”。与此同时，马尔科夫过程的成立需要满足以下两条假设：

有限的状态空间（finite state space）：马尔科夫过程的状态空间是有限的。这意味着对于马尔科夫过程 $(s_0, s_1, s_2, \ldots)$，任意一个状态空间 S 且 $|S| < \infty$，使得对于所有可能的马尔科夫过程的实现，都有 $s_i \in S,\ i = 1,2,\ldots$。

状态转移概率（stationary transition probabilities）不变：状态转移概率与时间无关。数学上的表示为：

$$P(s_i = s'|s_{i-1} = s) = P(s_j = s'|s_{j-1} = s), \forall s, s' \in S, \forall i, j = 1,2,\ldots \quad (2\text{-}1)$$

满足以上假设的马尔科夫过程有时也被称作马尔科夫链（Markov chain）。对于马尔科夫过程，这些假设使得我们可以用一个矩阵来描述状态转移：这个矩阵为转移概率矩阵（transition probability matrix）P，其大小为 $|S| \times |S|$，第 (i,j) 个元素为 $P_{ij} = P(j|i)$，这里 i 和 j 表示状态（随意排序）。注意，P 的元素是非负的，并且每行的和为 1。

因此，我们可以用 (S,P) 来定义一个马尔科夫过程：

S：有限的状态空间；

P：详细表述 $P(s'|s)$ 的状态转移概率模型。

2.1.3 马尔科夫奖励过程（MRP）

马尔科夫奖励过程[33]是一个马尔科夫过程，并且有特定的奖励函数（reward function）和衰减系数（discount factor），通常用 (S,P,R,γ) 表示：

S：有限的状态空间；

P：状态转移模型；

R：将状态映射到奖励（实数）的奖励函数，即 $R: S \to R$；

γ：衰减系数，范围为 $[0,1]$。

对于状态 $s \in S$，奖励的期望 $R(s)$ 可以定义为：

$$R(s) = E[r_0 | s_0 = s] \tag{2-2}$$

其中奖励函数表示从当前状态 s 转移到后续状态 s' 时，所获得的奖励值。因此对于马尔科夫过程 $(s_0, s_1, s_2, \ldots)$，每次状态转移 $s_i \to s_{i+1}$ 都会获得一个奖励值 r_i（对于所有 $i = 0,1,\ldots$）。所以，马尔科夫奖励过程可以表示为 $(s_0, r_0, s_1, r_1, s_2, r_2, \ldots)$。

2.1.4 马尔科夫决策过程（MDP）

马尔科夫决策过程继承了马尔科夫奖励过程的基本结构，在 MDP 中，智能体可以在每个状态下采取某些动作。MDP 可以用 (S, A, P, R, γ) 来表示：

S：有限的状态空间；

A：有限的动作空间；

P：描述 $P(s'|s, a)$ 的状态转移概率模型；

R：将状态-动作映射到奖励的奖励函数，即 $R: S \times A \to R$；

γ：衰减因子 $\gamma \in [0,1]$。

在动力学的基本模型中，包含了状态空间 S 和动作空间 A，我们将这两个空间视为有限的。智能体在 i 时刻从状态 s_i 开始，从动作空间中选择一个动作 a_i，得到一个奖励值 r_i 并转移到下一个状态 s_{i+1}。因此，MDP 中的一个过程片段可以被描述为 $(s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots)$。

在马尔科夫过程和马尔科夫奖励过程中，状态转移概率仅是下一状态以及当前状态的函数，而在 MDP 中，在 i 时刻的状态转移是下一状态 s_{i+1}、当前状态 s_i 以及动作 a_i 的函数，写作 $P(s_{i+1}|s_i, a_i)$。我们仍假设状态转移概率不变，在 MDP 中转移矩阵可以写为：

$$P(s_i = s' | s_{i-1} = s, a_{i-1} = a) = P\left(s_j = s' \middle| s_{j-1} = s, a_{j-1} = a\right)$$

$$\forall s, s' \in S, \forall i, j = 1,2,\ldots, \mathrm{a} \in \mathrm{A} \tag{2-3}$$

在 MDP 中，在 i 时刻的奖励 r_i 依赖于 s_i 和 a_i，而在马尔科夫奖励过程中，奖励只依赖于当前状态。MDP 中的奖励可以是随机性的或确定性的，但就像马尔科夫奖

励过程那样，我们假设奖励是确定性的，并且唯一相关的量是奖励期望，对于特定的状态 s 和动作 a，我们将这个奖励值的期望记为 $R(s,a)$：

$$R(s,a) = E[r_i|s_i = s, a_i = a],\ \forall i = 0,1,\ldots \quad (2\text{-}4)$$

2.1.5 时序差分学习（TD Learning）

TD Learning（Temporal-Difference Learning） 时序差分学习指的是一类无模型的强化学习方法，它是从当前价值函数估计的自举过程中学习的。这些方法从环境中取样，如蒙特卡洛方法，并基于当前估计执行更新，如动态规划方法。

动态规划利用引导来帮助我们在只需要一步备份的条件下去获取价值估计。另一方面，蒙特卡洛对多条轨迹进行采样，使我们不需要知道模型也可以估计价值。而时序差分学习则将引导和采样相结合，为了理解将采样和引导的结合过程，让我们先了解一下蒙特卡洛的增量更新方式：

$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha(G_t - V^{\pi}(s_t)) \quad (2\text{-}5)$$

G_t 是在状态 s_t 从时间步 t 直到终止执行某个策略所获得的回报。我们用 $r_t + \gamma V^{\pi}(s_{t+1})$ 替代 G_t，这里 r_t 是在时间步 t 的奖励的采样，$V^{\pi}(s_{t+1})$ 是当前对下一状态的值的估计。替换之后，我们得到了时间差分学习（TD-learning）更新：

$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha(r_t + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)) \quad (2\text{-}6)$$

完整的时序差分学习算法如算法 2-1 所示。

算法 2-1 TD Learning 来验证策略 π

1: **procedure** TD Learning (学习率 α, 轨迹数 n)
2: 对于所有的状态 s, $V^{\pi}(s) \rightarrow 0$
3: **while** n > 0 **do**
4: 开始的片段 E 在状态 s
5: **while** n > 0 并且片段 E 没有终止 **do**
6: $a \leftarrow action$ 在状态 s 在策略 π 下
7: 执行动作 a 在 E 时 ，奖励值为 r, 下一个状态 s'
8: $V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha(r_t + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t))$
9: $s \leftarrow s'$
10: **return** V^{π}

2.1.6 Q-learning

Q-learning本质上来说是基于TD(0) 实现的算法，Q-learning的目的是学习特定state、特定 action 下的价值。算法实现上，Q-learning 以 state 为行、action 为列，建立一个 Q-Table，通过更新每个动作时获得的奖励值来更新 Q-Table。Q-learning 是 off-policy 的算法。off-policy 是指行动策略和评估策略不是同一个策略。Q-learning 中行动策略是 ε-greedy 策略，而更新 Q 表的策略是 greedy 策略，其按照如下公式更新 Q 值：

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha_t[r_t + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t)] \qquad (2\text{-}7)$$

完整的 Q-learning 如算法 2-2 所示。

算法 2-2 Q-learning 通过 $\in -greedy$ 的方式进行探索

procedure Q-Learning $(\in, \alpha, \gamma)$
    初始化 Q(s, a) 对于所有的 $s \in S$，$a \in A$ 除非 $Q(terminal,\cdot) = 0$
    $\pi \leftarrow \epsilon - greedy$ 策略对于 Q
    for 每一个片段 **do**
        将 s_1 设置为初始状态
        $t \leftarrow 1$
        loop 直到片段终止
            从策略 $\pi(s_t)$ 中采样得到动作 a_t
            执行动作 a_t 并且得到奖励值 r_t，下一个状态 s_{t+1}
            $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha_t[r_t + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t)]$
            $\pi \leftarrow \epsilon - greedy$ 策略对于 Q (策略提升)
            $t \leftarrow t + 1$
    return Q, π

2.1.7 深度强化学习 (Deep Q-learning)

自从 2013 年 DQN 问世之后，各种深度强化学习算法[35]层出不穷。但这里我们主要介绍 DQN，其他的深度强化学习算法将在第 3 章第二节进行介绍。

受 Q-Table 大小的限制，在未使用神经网络之前，我们只能在较小的状态和动作空间中使用到强化学习。而神经网络这个强大的映射函数，用来代替 Q-Table 实在是最适

合不过了。实验结果也表明，DQN 在 Atari 2600 的 49 个游戏中达到了相当于专业的游戏测试员的水平[25]。

图 2-2 展示了 DQN 的结构，图中的网络将 Atari 游戏环境的预处理像素图作为输入，为每个可行的动作赋予一个 Q 值，将这些 Q 值作为一个向量输出。预处理的像素输入代表了游戏状态 s，单个的输出单元表示动作 a 的 $\hat{q}$ 函数。总的来说，$\hat{q}$ 可以被记为 $\hat{q}(s,w) \in R^{|A|}$，简单起见，我们仍用 $\hat{q}(s,a,w)$ 来表示对 (s,a) 的状态-行为值估计。

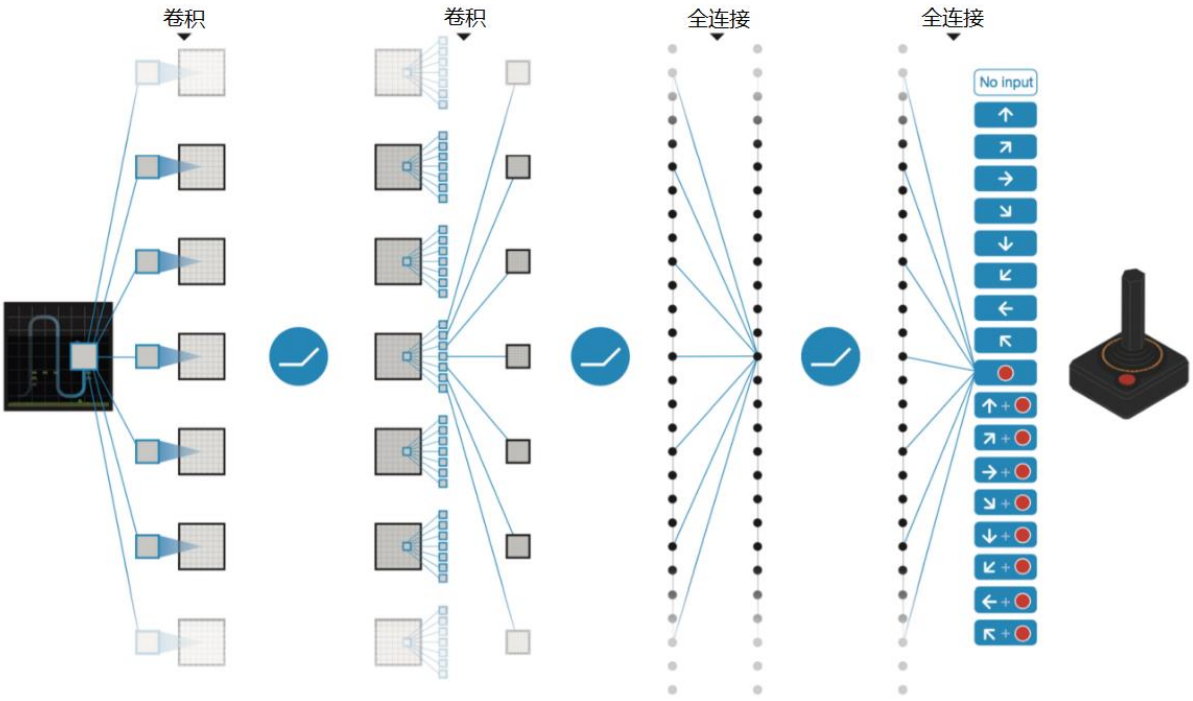

图 2-2 深度 Q 网络：网络的输入为一张 84×84×4 的预处理过的图片，后面接三个卷积层和两个全连接层，每个动作都对应一个输出，每个隐藏层都接一个非线性激活函数（ReLU）

由于没有理论的保证，而且学习和训练往往很不稳定，因此过去常常避免使用大型的深度神经网络来近似状态-行为值函数。为使用大型非线性近似函数和大规模在线 Q-learning，DQN 引入了两个主要变化：使用经验回放（experience replay）和一个单独的目标网络（target network）。

经验回放（Experience Replay）：智能体在每个时间步的经历（或状态转移）$e_t = (s_t, a_t, r_t, s_{t+1})$ 被存储在一个固定大小的数据集（或回放存储器，replay buffer）中：$D_t = e1, \ldots, et$，回放存储器保留着最新的 $k = 1million$ 次经历。通过以最小批次数据采样计算梯度来更新 Q-network，最小批次中的每个状态转移样本都是从经验存储池中随机均匀采样得到的：$(s,a,r,s') \sim U(D)$。对于标准的在线 Q-learning，可以提高数据的使

用效率（greater data efficiency）并且去除样本间的相关性（removing sample correlations）。但经验回放也存在以下缺点：回放存储器并不区分重要的或能提供有用信息的状态转移；由于存储容量固定，最旧的状态转移总是会被最新的状态转移覆盖，而最新的状态转移并不一定比旧的状态转移更有效。

目标网络（Target Network）：为了进一步提高学习的稳定性，在 Q-learning 更新时，DQN 采用了一个独立的目标网络来生成目标 y_i（算法 2-3 第 12 行）。具体来说，通过从在线网络 $\hat{q}(s,a,w)$ 复制参数值 $w^- = w$，每过 C 次时间步，更新一次目标网络 $\hat{q}(s,a,w^-)$，在接下来的 C 次更新中，目标网络保持不变的同时更新 y_i。这项改动使算法相对标准的在线 Q-learning 而言更加稳定。

本质上，Q-network 是通过降低均方差来学习的，均方差公式为：

$$J(w) = E_{(s_t,a_t,r_t,s_{t+1})}[\left(y_t^{DQN} - \hat{q}(s_t,a_t,w)\right)^2] \tag{2-8}$$

这里 y_t^{DQN} 为提前一步学习目标（one-step ahead learning target）：

$$y_t^{DQN} = r_t + \gamma \max_{a'} \hat{q}(s_{t+1},a',w^-) \tag{2-9}$$

w^- 表示目标网络的参数，在线网络参数 w 采用梯度下降进行更新，数据来源于历史的状态转移 (s_t,a_t,r_t,s_{t+1})。

算法 2-3 deep Q-learning

1: 使用固定大小初始化经验池 D

2: 使用随机权重 w 初始化动作-价值函数 $\hat{q}$

3: 使用权重 $w^- = w$ 初始化目标动作-价值函数 q

4: **for** 片段 $m = 1,\dots,M$ **do**

5: 观察到初始的画面 x_1，并对其进行预处理得到 s_1

6: **for** 时间步 $t = 1,\dots,T$ **do**

7: 选择动作 $a_t = \begin{cases} random\ action & with\ probability\ \in \\ \arg\max_a \hat{q}(s_t,a,w) & otherwise \end{cases}$

8: 在虚拟环境中执行动作 a_t 并且得到奖励值 r_t 和图片 x_{t+1}

9: 初始化 s_t，x_{t+1} 来获取 s_{t+1}，并且将轨迹 (s_t,a_t,r_t,s_{t+1})存储在 D 中

10: 从 D 中随机采样小批量轨迹 (s_j,a_j,r_j,s_{j+1})，数据量为 N

11: 如果片段在 $j+1$ 处终止，则设 $y_j = r_j$;

12: 否则设 $\mathrm{y_j} = r_j + \gamma \max_{a'} \hat{q}(s_{t+1},a',w^-)$

13: 使用随机梯度下降来计算 $J(w)=\frac{1}{N}\sum_{i=1}^{N}\left(\mathrm{y_j}-\hat{q}(s_j,a_j,w)\right)^2$

14: 每经过 C 步，将重置 $w^{-}=w$

2.2 量化交易

量化交易[12]是指借助现代统计学和数学的方法,利用计算机技术来进行交易的证券投资方式。量化交易从庞大的历史数据中海选能带来超额收益的多种“大概率”事件以制定策略，用数量模型验证及固化这些规律和策略,然后严格执行已固化的策略来指导投资，以求获得可以持续的、稳定且高于平均收益的超额回报。

量化交易起源于上世纪七十年代的股票市场,之后迅速发展和普及,尤其是在期货交易市场,程序化逐渐成为主流。有数据显示，国外成熟市场期货程序化交易已占据总交易量的 70%-80%，而国内则刚刚起步。手工交易中交易者的情绪波动等弊端越来越成为盈利的障碍，而程序化交易天然而成的精准性、100%执行率则为它的盈利带来了优势。

同时，量化交易也有如下的优势：

1.严格的纪律性

量化交易有着严格的纪律性,这样做可以克服人性的弱点,如贪婪、恐惧、侥幸心理,也可以克服认知偏差。一个好的投资方法应该是一个“透明的盒子”。我们的每一个决策都是有理有据的，特别是有数据支持的。如果有人质问我，某年某月某一天， 你为什么购买某支股票的化，我会打开量化交易系统，系统会显示出当时被选择的这只股票与其他的股票相比在成长面上、估值上、资金上、买卖时机上的综合评价情况，而且这个评价是非常全面的,比普通投资者拍脑袋或者简单看某一个指标买更具有说服力。

2.完备的系统性

完备的系统性具体表现为“三多”。首先表现在多层次，包括在大类资产配置、行业选择、 精选个股三个层次上我们都有模型;其次是多角度,量化交易的核心投资思想包括宏观周期、市场结构、估值、成长、盈利质量、分析师盈利预测、市场情绪等多个角度,也就是对海量数据的处理。人脑处理信息的能力是有限的，当一个资本市场只有 100 只股票[36],这对定性投资基金经理是有优势的，他可以深刻分析这 100 家公司。但在一个很大的资本市场， 比如有成千上万只股票的时候,大的定量化交易的信息处理能力能反映它的优势，能捕捉更多的投资机会，拓展更大的投资机会。

3.妥善运用套利的思想

量化交易正是在找估值洼地,通过全面、系统性的扫描捕捉错误定价、错误估值带来的机会。定性投资大部分时间在琢磨哪一个企业是伟大的企业，那个股票是可以翻倍的股票;与定性投资不同,化交易大部分精力花在分析哪里是估值洼地，哪一个品种被低估了，买入低估的，卖出高估的。

4.靠概率取胜

这表现为两个方面，一是量化投资不断的从历史中挖掘有望在未来重复的历史规律并且加以利用。二是在股票实际操作过程中，运用概率分析,提高买卖成功的概率和仓位控制。

第 3 章 基于深度强化学习的量化投资策略的构建

3.1 构建股票交易环境

在训练一个深度强化学习的交易代理之前，我们需要搭建模拟真实交易的环境[37]，让代理在与环境的交互中进行学习。在实际交易过程中，需要考虑各种信息，例如历史股价、当前持有的股票、技术指标等。我们的交易代理需要通过环境获取这些信息，并采取上一章中定义的操作，我们通过 Tushare API 获取交易的数据，使用 OpenAI gym 来构建我们的环境，以此来训练我们的代理[13]。

我们使用一个连续的动作空间来对多个股票的交易进行建模，并使用上证 50 成分股作为我们交易的股票池。同时，在构建环境的过程中参考了 FinRL[23]的构建步骤。

3.1.1 状态空间

我们使用一个包含 17 部分信息的 801 维向量来表示多个股票交易环境[38]的状态空间：$[b_t, p_t, h_t, M_t, R_t, C_t, X_t]$。每个组件的定义如下（n 表示交易的股票数量）：

- 现金余额 $b_t \in R_+$：在 t 时刻，账户中剩余的现金。
- 持有的股份 $h_t \in Z_+^n$：每只股票当前的持仓量。
- 成交量 $v_t \in R_+^n$：可交易的时间段内股票成交总量。
- 收盘价 $p_t \in R_+^n$：每支股票调整后的收盘价。
- 20 日移动平均线 $MA_t^{20} \in R_+^n$：前 20 个交易日收盘价的平均值。
- 60 日移动平均线 $MA_t^{60} \in R_+^n$：前 60 个交易日收盘价的平均值。
- 120 日移动平均线 $MA_t^{120} \in R_+^n$：前 120 个交易日收盘价的平均值。
- 成交量的 20 日移动平均线 $VOLMA_t^{20} \in R_+^n$：前 20 个交易日成交量的平均值。
- 成交量的 60 日移动平均线 $VOLMA_t^{60} \in R_+^n$：前 60 个交易日成交量的平均值。
- 成交量的 120 日移动平均线 $VOLMA_t^{120} \in R_+^n$：前 120 个交易日成交量的平均值。
- BOLL 指标上轨线 $BOLL_t^{UP} \in R_+^n$：BOLL 指标属于路径指标，股价波动在上限和下限的区间之内，在正常情况下，股价应始终处于股价信道内运行，若股价突破上轨线，则表明股价处于超买状态，提醒观察者可以适当减仓
- BOLL 指标下轨线 $BOLL_t^{DN} \in R_+^n$：在正常情况下，股价应始终处于股价信道内运行，若股价突破下轨线，则表明股价处于超卖状态，提醒观察者可以适当加仓

- RSI $R_t \in R_+^n$：RSI 量化了近期价格变化的程度。如果价格在支撑线附近，这表明该股票是超卖的，我们可以执行买入操作。如果价格在阻力线附近，这表明该股票是超买的，我们可以执行卖出操作。
- MACD $M \in R^n$：使用收盘价计算得到。MACD 称为收敛和发散移动平均线。MACD 指标是 DIF，DEA 和 MACD 的组合。DIF 由快速趋势线表示。DEA 由慢速趋势线表示。MACD 由 DIF 和 DEA 下方的矩形直方图表示。

3.1.2 动作空间

对于某支股票，动作空间的定义为 $\{-\mathrm{k},\ldots,-1,0,1,\ldots,\mathrm{k}\}$，其中 k 和 $-k$ 表示我们可以购买和出售的股份数量，$k \le h_{\max}$，$h_{\max}$ 是一个自定义的参数，定义为每次可购买的最大股份数。因此，整个动作空间的大小为 $(2k+1)^n$。 因为 RL 算法 A2C 和 PPO 直接使用高斯分布输出策略的分布，需要进行归一化处理[14]，所以动作空间被归一化为 $[-1,1]$。

3.2 强化学习算法

我们使用五种基于 actor-critic 的算法[34]来实现我们的交易代理。这五种算法分别是 A2C、DDPG、PPO、SAC 和 TD3。以下是五种算法的介绍。

3.2.1 Advantage Actor Critic (A2C)

A2C[15]是典型的 actor-critic 算法。A2C 利用优势函数来减小策略梯度的方差，从而改进策略梯度的更新。critic 网络不只估计 value 函数，同时也用来估计 advantage 函数。因此，对动作的评估不仅取决于动作有多好，而且还要考虑它能变得多好。从而减小了 policy 网络的方差，提升了模型的鲁棒性。

A2C 使用同一个智能体的副本的不同的数据样本更新梯度。每个智能体是独立的，但和同一个环境进行互动。每次进行迭代时，当所有的智能体计算完它们的梯度后，A2C 使用一个协调器将所有智能体梯度的平均值传递到一个全局网络中，使用全局网络来更新 actor 和 critic 网络。全局网络的存在增加了训练数据的多样性。同步梯度更新的方式也更快，更适合对大批量数据进行处理。同时，由于 A2C 的稳定性，A2C 是一个非常棒的股票交易算法。

A2C 的目标函数为：

$$\nabla J_\theta(\theta) = E[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A(s_t,a_t)] \quad (3\text{-}1)$$

其中，$\pi_\theta(a_t|s_t)$ 为策略网络，$A(s_t,a_t)$ 为优势函数，同时可以被写为：

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t) \quad (3\text{-}2)$$

或者，

$$A(s_t, a_t) = r(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1}) - V(s_t) \quad (3\text{-}3)$$

3.2.2 Deep Deterministic Policy Gradient (DDPG)

DDPG[16]也可以用来最大化投资回报。DDPG 整合了 Q-leaning 和策略梯度框架，并使用神经网络作为函数逼近器进行学习。DQN 通过 Q 值映射进行间接学习，存在梯度爆炸的问题，而 DDPG 通过策略梯度直接从策略中学习。为了更好地适应连续动作的环境，DDPG 提出了确定性地将状态映射到动作的方式。

在每次更新时，DDPG 智能体在 s_t 处执行动作 a_t，到达 s_{t+1}，并得到奖励值 r_t。路径元组 (s_t, a_t, s_{t+1}, r_t) 便存储在 replay buffer R中。更新时从 R 中提取 N 组数据，用来更新 Q-value y_i：

$$y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1}|\theta^{\mu'}, \theta^{Q'}\right)\right), i = 1, \ldots, N \quad (3\text{-}4)$$

然后通过最小化损失函数 $L(\theta^Q)$ 来更新 critic 网络。同时，损失函数为 target critic 网络 Q' 的输出与 critic 网络 Q 的输出之间的期望差值：

$$L(\theta^Q) = E_{s_t, a_t, r_t, s_{t+1} \sim buffer}[(y_i - Q(s_t, a_t|\theta^Q))^2] \quad (3\text{-}5)$$

DDPG 非常适合用于处理连续动作空间，因此非常适用于股票交易。

3.2.3 Proximal Policy Optimization (PPO)

使用 PPO[17]控制策略梯度更新，可以确保新策略与之前的策略不会有太大差异。PPO 试图通过在目标函数中引入一个剪切函数来简化 Trust Region Policy Optimization (TRPO)的计算过程。

假设新老策略的比值为：

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \quad (3\text{-}6)$$

PPO 裁剪后的目标函数则为：

$$J^{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta)\hat{A}(s_t, a_t), clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}(s_t, a_t))] \quad (3\text{-}7)$$

其中 $r_t(\theta)\hat{A}(s_t, a_t)$ 为策略梯度目标函数，$\hat{A}(s_t, a_t)$ 为估计的优势函数。clip 函数 $clip(r_t(\theta), 1-\epsilon, 1+\epsilon)$ 将比值 $r_t(\theta)$ 剪切至 $[1-\epsilon, 1+\epsilon]$ 之间。PPO 的目标函数取剪切目标和正常目标之间的最小值，以此来约束策略的变化，让策略的变化不至于过

大。因此，PPO 通过限制每个训练步骤的策略更新，来提高策略网络训练的稳定性。我们选择 PPO 来交易股票，也是因为它很稳定，训练和实现也很简单。

3.2.4 Soft Actor Critic (SAC)

SAC[18]是一种优化随机策略的 off-policy 方法，它结合了随机策略方法和 DDPG-style 方法。它不能算是 TD3 的直接改进算法，但它使用了很多 TD3(Twin Delayed DDPG)的 trick，比如 clipped double-Q，并且由于 SAC 策略固有的随机性，它还受益于 target policy smoothing 之类的 trick。

SAC 的一个很重要的 feature 是 entropy regularization。这个算法的目标是最大化期望回报的同时，尽可能的最大化策略的熵（即使策略随机性更大）。这种结构很像探索与利用的 trade-off，因为更大的熵有助于更多的探索，同时也有助于防止策略过早收敛局部最优。

为了解释 SAC，必须先介绍一些熵正则化强化学习（entropy-regularized reinforcement learning）的设定，其中的值函数与我们平时看到的其他强化学习算法的值函数有一些区别，令随机变量 x 的概率密度函数为 P，x 的熵 H 可以表示为：

$$H(P) = \underset{x\sim P}{E}[-log\ P(x)] \tag{3-8}$$

在熵正则化强化学习中，智能体在每个 step 都会获得与当前的策略熵成正比的正向 reward 奖励，因此 RL 问题可以改写为：

$$\pi^* = arg\max_{\pi} \underset{\tau\sim\pi}{E}[\sum_{t=0}^{\infty}\gamma^t(R(s_t,a_t,s_{t+1}) + \alpha H(\pi(\cdot|s_t)))] \tag{3-9}$$

Q^π 的 Bellman 方程可以写为：

$$Q^\pi(s,a) = \underset{\substack{s'\sim P\\ a'\sim\pi}}{\mathrm{E}}\left[R(s,a,s') + \gamma\left(Q^\pi(s',a') + \alpha H\left(\pi(\cdot|s')\right)\right)\right]$$

$$= \underset{s'\sim P}{\mathrm{E}}[R(s,a,s') + \gamma V^\pi(s')] \tag{3-10}$$

3.2.5 Twin Delayed DDPG (TD3)

TD3[19]在 DDPG 的基础上进行了一系列的改进。主要解决了两个问题，一个是 overestimation bias，另一个是 high variance。

在离散动作空间的强化学习中，典型的算法如 DQN，值函数往往会被过高估计。这是因为，我们对于 Q 函数的估计会有误差，在取最大化 Q 值的时候，会高于真实的最大 Q 值。由于过高的估计偏差，这种累积的错误会导致任意较差的状态被高估，从而导

致次优的策略更新和发散的行为。

TD3 采用了 Double Q-learning 中的方法，使用一对独立的神经网络估计 critic。但是这种方法虽然能解决 bias 问题，但是同时也会带来高方差。为了解决这个问题，文章提出了一个 clipped Double Q-learning 版本，来降低方差：

$$y_1 = r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \pi_{\phi_1}(s')) \quad (3\text{-}11)$$

同时 TD3 还使用三个技巧来降低方差：

1、使用 target networks，这是从 DQN 开始就一直使用的技巧；

2、使用 Delaying policy updates 来解决值函数和策略耦合问题；

3、提出了一种正则化方法解决 determinstic policy 可能 overfitting 的问题。

更新时和 DDPG 比较类似，这里便不再过多赘述。

3.3 奖励函数的设置

强化学习中，奖励值的设定非常重要，奖励值的设定，将直接影响到智能体的学习效果。在股票交易中，我们可以把智能体取得的累计收益率作为奖励值，这非常合理。但为了让收益更加稳健，我们还可以在奖励值中加入当前回撤率，作为负的奖励值，也可以说是惩罚值。所以，在设定奖励值函数的时候，我采用的方案为：将累计收益率 - 当前回撤率作为奖励值。

按理来说我们也可以将 Sharpe 比率或者 Omega 比率加入到奖励函数中，因为他们非常适合用于评价智能体对收益和风险的把控。但在实际的训练过程中却是没法实现的，因为在刚开始训练的阶段波动较小，Sharpe 比率和 Omega 比率会接近于无限大，那我们过段时间再把这俩指标加入到奖励值函数中不就可以了吗？这么做按道理来说是不可行的，因为我们需要保证在训练过程中奖励值函数的一致性。

那么，收益曲线的斜率可以考虑吗？如果我们将收益曲线的斜率添加到奖励值函数中，那么当获得收益的速度过快时，将会有非常大的奖励值或者惩罚值，这会对极端情况进行较大的奖励和惩罚。如此，会加大智能体的风险偏好，将其暴露在风险中，会让训练后的智能体缺少我所需要的抵御风险的能力，这不是我想要的结果。所以，在设置奖励值函数的时候，我并没有考虑收益曲线的斜率。

是否也可以对持有的现金进行管控呢？例如，如果智能体持有较多现金时，对其进行惩罚，让智能体保证较大的持仓量，因为长期来看，股市的回报是值得期待的。这确实是一个可以考虑的点，或许在之后的环境构建中可以将其考虑进来。此次奖励值

函数的设置过程中未将其考虑的主要原因是，想知道智能体在不对其进行约束的情况下，也有很好的现金管控能力。

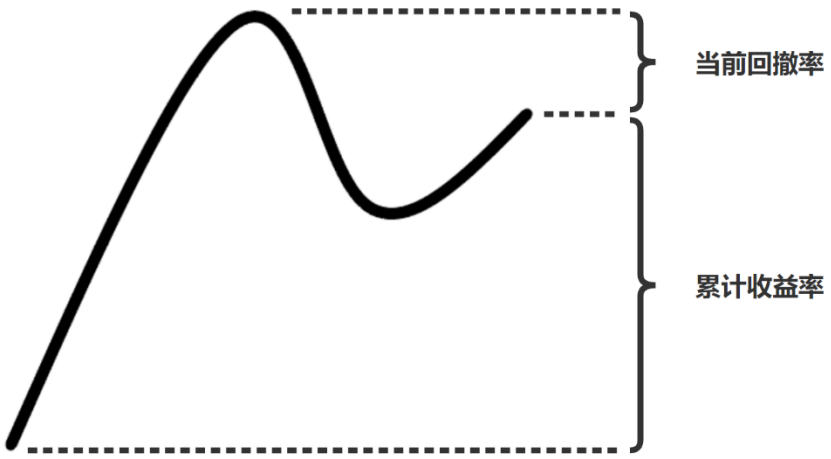

图 3-1　奖励值函数设置示意图

如下是累计收益率和当前回撤率的定义：

3.3.1　累计收益率

累计收益率（Cumulative Return Ratio）的概念十分简单，表示的就是持有资产期间，所获得或者损失的资产的比率。其计算公式为：

$$\mathrm{CumulativeReturnRatio} = \frac{\mathrm{R_t}}{A_s} \tag{3-12}$$

其中，$\mathrm{R_t}$ 表示持有资产期间资金的变动，A_s 表示为初始资金量。

3.3.2　当前回撤率

当前回撤率（Current Drawdown Ratio）是指当前资产总值和资产最大值之间的回撤比率。当前回撤率可以用来衡量智能体管控风险的能力。

第 4 章 算法结果评估

在本节中，我们将对第 3 章中所提出方案的性能进行评估。在训练完成之后，我们将对五个智能体进行回测。表 4-1 的结果表明，在 Sharpe 比率和 Omega 比率方面，PPO 智能体比其他智能体和上证 50 指数有着更好的表现，如果需要构建一个稳健的投资组合的话，PPO 智能体将会是一个不错的选择。

可以在 Github 上查看到论文的代码，仓库名为 sunnyswag/RL_in_Stock。

4.1 系统设计

在书写量化交易策略研究的系统时，参考了 FinRL[23]的设计理念，为了保证其拓展性和可维护性，我将其设计架构分为三层，底层为 Utils 层，完成的是一些基础事件的操作，中间层为强化学习 Algorithms 层，使用的为 Stable Baseline 所实现的强化学习算法。最顶层为 Learn 层，对底层的实现进行调用，完成整个算法训练和回测的流程。在最顶层，我们可以通过设置输入的参数，非常方便的调整训练和测试的算法。整个结构的示意图如图 4-1 所示。接下来我将会对部分细节进行简要的介绍。

Learn	data		train		trade	
Algorithms	A2C	DDPG	PPO	SAC	TD3	
Utils	pull_data	preprocessors	env	model	config	backtest

图 4-1 系统结构示意图

4.1.1 股票数据预处理

我选择的是上证 50 指数的 50 支成分股(2021 年 1 月 1 日)作为进行交易的股票池。回测时使用的是 2009 年 1 月 1 日至 2020 年 5 月 8 日的历史数据。股票数据通过 Tushare API 进行下载。下载完成之后，我们对股票数据进行数据预处理的操作，数据预处理的主要内容为添加技术指标，并把技术指标添加到状态空间中，技术指标的定义在 3.1.1 有介绍。

状态空间设置完成之后，我们再对状态空间进行拆分，分为训练状态空间和测试状态空间。整个状态空间的拆分方式，如图 4-2 所示。训练状态空间的范围为 2009 年 1 月 1 日至 2019 年 1 月 1 日，测试状态空间的范围为 2019 年 1 月 1 日至 2021 年 1 月 1 日。五个算法训练完成之后，我们再分别对其进行回测。

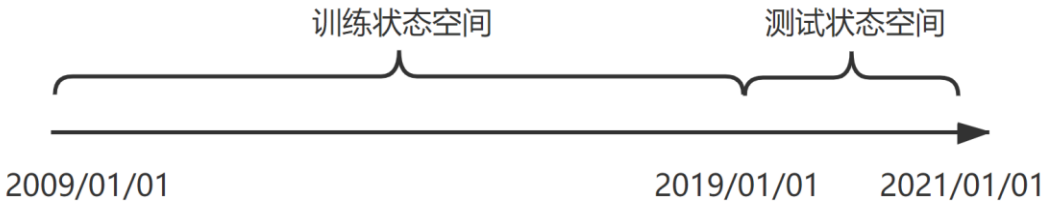

图 4-2　股票数据分割

4.1.2　股票交易环境的构建

考虑到股票自动交易任务的随机性和交互性，可以将金融任务构建为马尔可夫决策过程（MDP）问题。训练过程包括监测股票价格变化，采取行动和计算奖励值，以使智能体优化其策略。随着时间的推移，智能体通过与环境的交互中不断学习，将得到最优的交易策略。交易环境基于 OpenAI Gym[13]框架进行构建，根据时间驱动模拟的原理，使用真实的历史市场数据来模拟真实的股票市场。目前的环境只使用了上证 A 股的指数，不过由于金融交易数据的差异性较小，使用其他市场的数据来构建环境也是可以直接使用的，如期货、其他股票交易市场等。

4.1.3　自动化回测

回测在性能评估中起着关键作用。我们最好使用自动化回测工具，因为它可以减少人为错误。在实现时，我使用的为 Quantopian pyfolio 代码库[31]对我们的交易策略进行回测。该软件包易于使用，由多个单独的图组成，这些图能让我们对交易策略性能有着更加全面的了解。同时，有较多的性能评估指标，可以用来帮助我们对策略的表现有着较为全面的了解。

4.1.4　数据-训练-回测流

在系统的顶层，即 Learn 层，我将系统的运行流程分为三个步骤，即：数据获取、训练和回测。按照这个步骤，我们会先将数据下载到本地并进行数据预处理的操作，再分别使用 5 个强化学习算法对获取到的数据进行训练，训练完会得到相应的强化学习模型。执行最后一个步骤时，我们通过强化学习模型在回测状态空间中进行回测，并得到相应的回测报告。在训练和回测的过程中，我使用的为每天的交易数据。当然，使用按

分钟、按月、按季度整合的交易数据也是可行的。

4.2 性能评估指标

回测完成之后，我们将每个算法的回测数据进行汇总，并对其进行评估。选用的评估指标除了比较常用的累计收益率以外，我们还添加了其他 5 个指标，这 6 个指标的介绍如下：

4.2.1 累计收益率

正如 3.3.1 所描述的累计收益率表示的就是持有资产期间，所获得或者损失的资产的比率。其计算公式为：

$$\text{CumulativeReturnRatio} = \frac{\mathrm{R_t}}{A_s} \tag{4-1}$$

其中，$\mathrm{R_t}$ 表示持有资产期间资金的变动，A_s 表示为初始资金量。

4.2.2 最大回撤率

最大回撤率（Max Drawdown Ratio）是指在回测阶段内任一历史时刻往后推，资产净值走到最低点时的收益率回撤幅度的最大值。最大回撤用来描述买入产品后可能出现的最糟糕的情况。

4.2.3 Omega 比率

2002 年，Shadwick 和 Keating 提出了一个新的业绩衡量指标，Omega 比率[21]。该指标考虑了收益率的整个分布信息，因此包括了所有高阶矩的信息。作为投资绩效主要的评级依据，Omega 比率在国外已经受到业界认可。Omega 比率的定义如下：

$$\Omega(\mathrm{r}) \triangleq \frac{\int_r^{\infty}(1-F(x))dx}{\int_{-\infty}^{r}F(x)dx} \tag{4-2}$$

其中，r 为指定的临界收益率，$F(x)$ 为收益率的累计分布函数。

这个公式的分子部分是上偏矩，分母部分是下偏矩，通常我们用定积分进行计算，而阈值 r 是我们设定的临界收益率。临界收益率用来区分收益或损失，高于临界收益率的，视为收益，低于临界收益率的，视为损失，也可以称为最低要求收益率。Omega 比率利用了收益率分布的所有信息，考虑了所有的高阶矩，刻画了收益率风险的所有特征。

可以通过图 4-3 来理解该指标的计算：

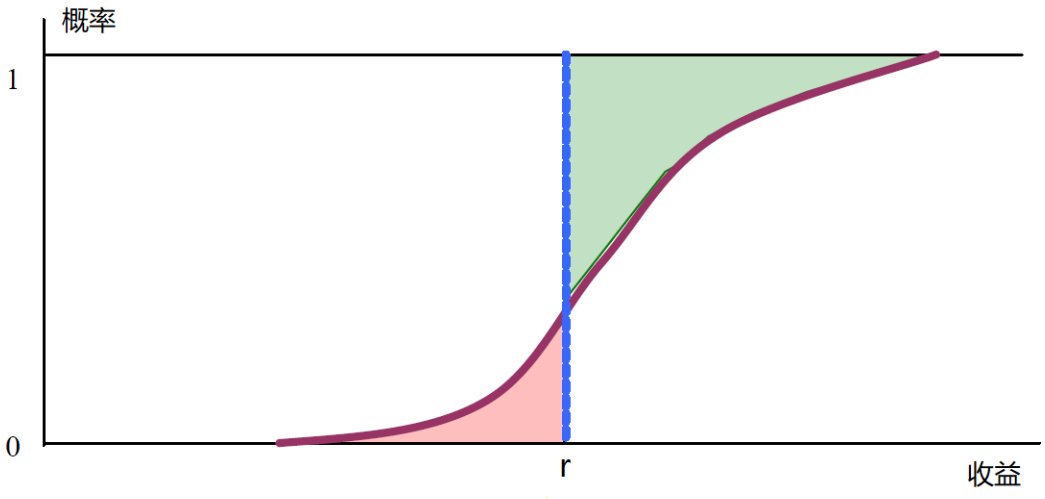

图 4-3　Omega 比率示意图

4.2.4　Sharpe 比率

1990 年度诺贝尔经济学奖得主威廉夏普(William Sharpe)以投资学最重要的理论基础 CAPM（Capital Asset Pricing Model, 资本资产定价模式)为出发，发展出名闻遐迩的夏普比率(Sharpe Ratio)[20]又被称为夏普指数, 用以衡量金融资产的绩效表现。

威廉夏普理论的核心思想是：理性的投资者将选择并持有有效的投资组合, 即那些在给定的风险水平下使期望回报最大化的投资组合，或那些在给定期望回报率的水平上使风险最小化的投资组合。解释起来非常简单，他认为投资者在建立有风险的投资组合时，至少应该要求投资回报达到无风险投资的回报，或者更多。Sharpe 比率的计算公式为：

$$SharpeRatio = \frac{E(R_P)-R_f}{\sigma_P} \qquad (4\text{-}3)$$

其中，R_P 为交易时间段内的投资回报，R_f 为交易时间段内的无风险回报，σ_P 为 R_P 的标准差。

4.2.5　年化收益率

年化收益率[22]仅是把当前收益率(日收益率、周收益率、月收益率)换算成年收益率来计算的，是-种理论收益率，并不是真正的已取得的收益率。例如日利率是万分之一， 则年化收益率是 3.65%（平年是 365 天)。

4.2.6 年化波动率

年化波动率即投资组合收益的年化标准差。

在这些指标中，累计收益率反映的是交易阶段结束时的回报。年化收益率是指投资组合在每年年底的回报率。年化波动率和最大回撤率可以用来衡量模型的控制风险的能力。Omega 比率和 Sharpe 比率是两个广泛使用的指标，它们将收益和风险结合在一起进行评估。

4. 3 智能体的性能评估

4.3.1 收益分析

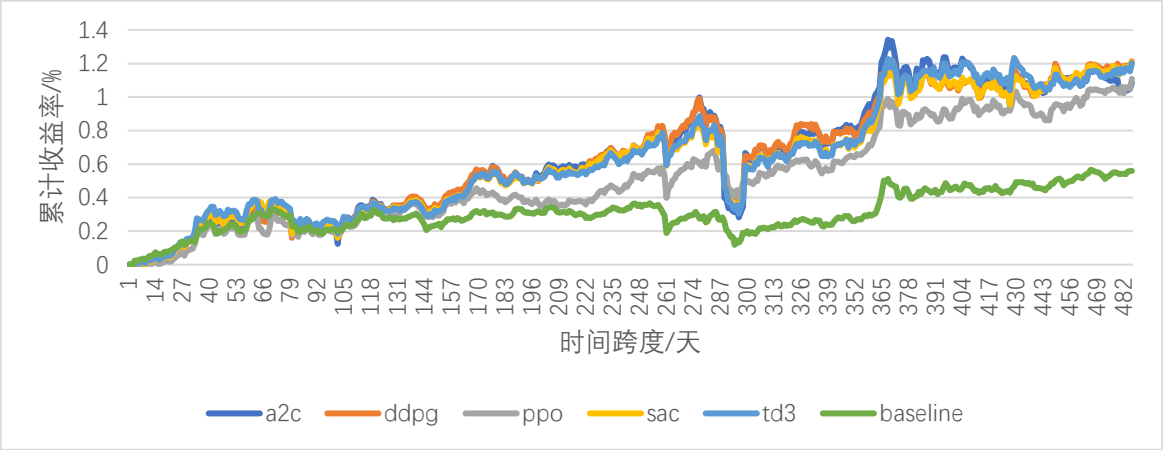

图 4-4 回测结果展示（时间跨度为 2019 年 1 月 1 日至 2021 年 1 月 1 日）

从表 4-1 和图 4-4 中可以看出，在回测阶段，DDPG 的累计收益率最大， 同时 DDPG 的最大回撤也是最小的，Sharpe 比率也并不高，可见 DDPG 智能体在执行一种比较激进的投资策略，信奉着风险越大，收益越高。PPO 虽然累计收益不高，但 Omega 比率和 Sharpe 比率却是这几个智能体中最大的。在控制回撤方面，它做的甚至比上证 50 指数还要好，而收益却是上证 50 的一倍左右。毫无疑问，PPO 智能体是最优秀的。

同时，我们也可以根据累计收益率，将这五个智能体分为两组，第一组为 PPO 和 A2C，第二组为 DDPG、SAC 和 TD3。第一组的累计收益率大概比第二组的少 10%。比较有趣的是，从算法实现的角度来看，这么分组也是合理的。第一组使用的是随机性策略，神经网络输出的是高斯分布的均值和方差，再根据输出的高斯分布进行动作的采样。第二组使用的是确定性策略，神经网络直接输出动作的值，再将输出的值加上一个标准正态分布的噪声后作为动作输入到环境中。或许，从结果中也可以看出随机性策略和确定性策略的不同，后者比前者更为激进。

4.3.2 市场崩盘下的表现

在回撤阶段，正好遇上了 2020 年 3 月的股市崩盘行情。在图 4-4 及图 4-7 中，我们可以看到所有智能体的资产都出现了较大的回撤，PPO 智能体的回撤比率最小，DDPG 智能体的回撤比率最大。不过值得庆幸的是，五个智能体和上证 50 指数在这之后都走出了一波不错的行情。如果可以的话，在熊市中我们可以选用较为保守的 PPO 智能体，而在牛市中使用较为激进的 DDPG 智能体。

4.3.3 和上证 50 指数比较

通过 4.3.1 的分析我们可以得出表现最好的智能体为 PPO 智能体，而 PPO 智能体在这六个评估指标上都完胜上证 50 指数。其他智能体在累计收益率、年化收益率和 Omega 比率上都比上证 50 指数要高，但最大回撤率都比上证 50 指数的要低，分别为 -35.83%、-31.45%、-29.24% 和 -30.69%。而在 Sharpe 比率上，只有 A2C 智能体比上证 50 指数要低，两者的 Sharpe 比率分别为 1.23 和 1.37。研究结果表明，五个智能体的表现比上证 50 指数都要好，说明强化学习在金融交易上的应用是可行的。

表 4-1 回测结果分析表（时间跨度为 2019 年 1 月 1 日至 2021 年 1 月 1 日）

性能评价指标	上证 50 指数	A2C	DDPG	PPO	SAC	TD3
累计收益率	58.98%	108.49%	121.26%	110.85%	120.61%	120.14%
最大回撤率	-18.22%	-35.83%	-31.45%	-16.75%	-29.24%	-30.69%
Omega 比率	1.29	1.31	1.34	1.36	1.35	1.35
Sharpe 比率	1.37	1.23	1.50	1.72	1.54	1.52
年化收益率	27.11%	46.37%	50.95%	47.23%	50.72%	50.55%
年化波动率	18.90%	36.25%	30.78%	24.28%	29.48%	29.89%

4.3.4 对 PPO 智能体的详细分析

根据 4.3.1~4.3.3 的分析结果，我们可以了解到，如果需要选择一个既稳定又高效的强化学习投资组合，PPO 智能体是最佳的选择。如此，接下来我们来对 PPO 智能体进行一个较为详细的分析。

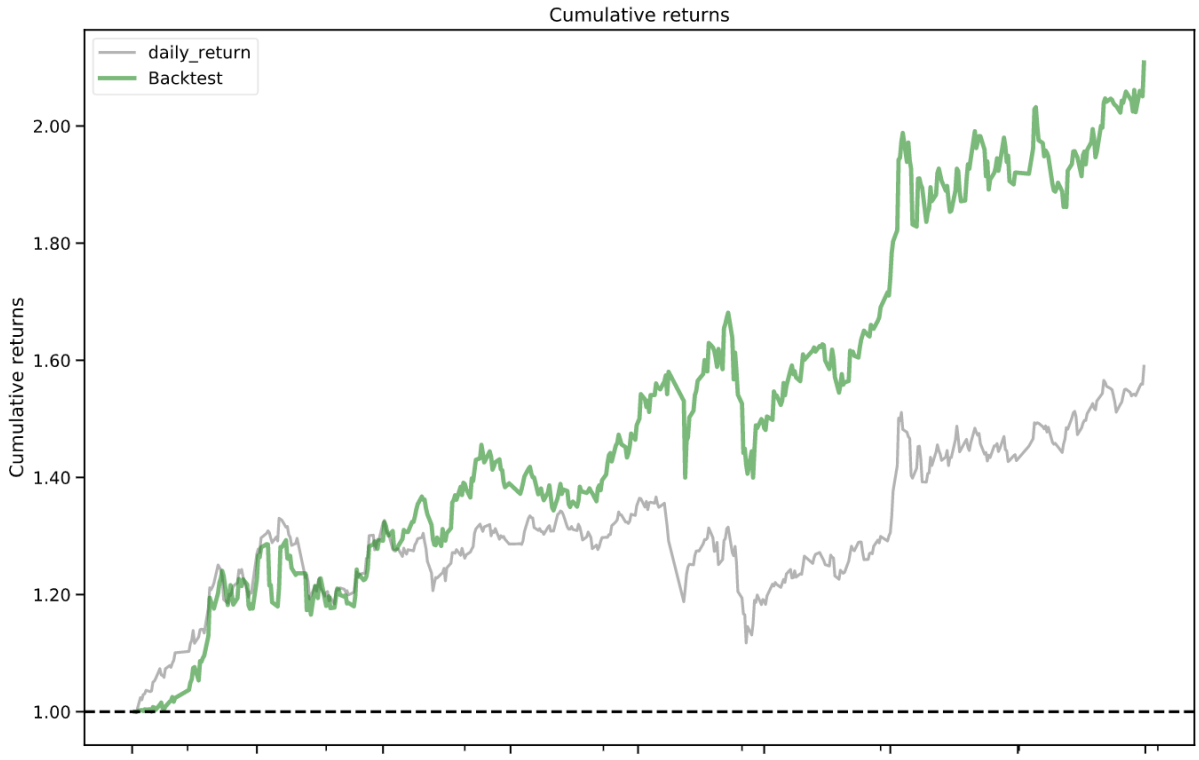

图 4-5 PPO 回测结果展示（时间跨度为 2019 年 1 月 1 日至 2021 年 1 月 1 日）

如图 4-5 所示，在回测时，PPO 智能体的表现并不是一直都优于上证 50 指数的，在回测的前期，PPO 智能体的收益大幅低于上证 50 指数，这和 PPO 智能体小心谨慎的特性有关。而小心谨慎，也帮助其在极端行情中有着优秀的回撤控制的能力，从而在收益上大幅超过上证 50 指数，最后取得了翻倍的好成绩。如果只从收益曲线来进行评定，PPO 智能体的表现，完全可以和一名优秀的交易员比肩。

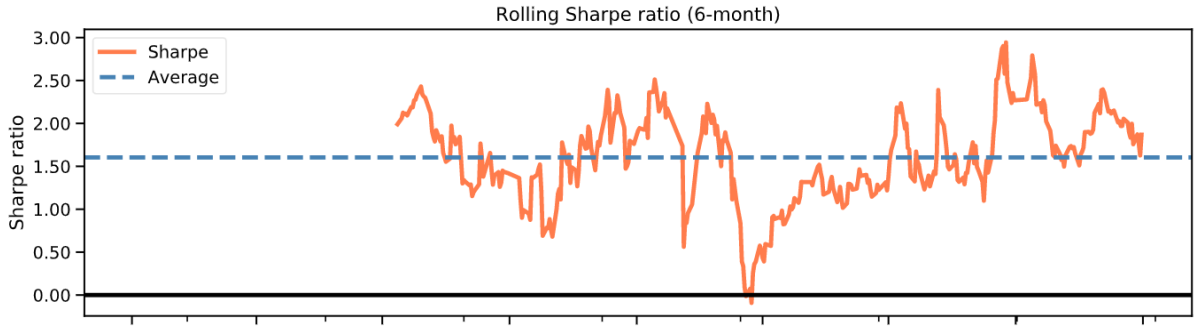

图 4-6 PPO 回测期间最后 6 个月的 Sharpe 比率走势

如图 4-6 所示，PPO 在这 6 个月期间的 Sharpe 比率并未出现为 0 的时间段。在这 6 个月期间，Sharpe 比率的平均数为 1.5 左右，这是一个非常健康的数值。而大多出情况下，PPO 智能体的 Sharpe 比率也在这个数值上下浮动，不得不再次对 PPO 智能体的稳定性发出感叹。

表 4-2 PPO 回撤比率分析表

回撤时期	回撤比率	开始日期	最低点日期	恢复日期	持续天数
1	16.75%	2020-03-06	2020-03-24	2020-06-24	79
2	11.43%	2020-01-23	2020-02-04	2020-02-18	19
3	9.82%	2019-04-22	2019-05-10	2019-06-28	50
4	8.38%	2020-10-14	2020-11-05	2020-12-02	36
5	8.25%	2019-04-16	2019-04-16	2019-04-22	11

如表 4-2 所示，PPO 的最大回撤为 16.75%，发生在 2020-03-06 至 2020-06-24 期间，最大回撤的位置为 2020-03-24，持续天数为 79 天。其他时期的回撤均未能和此次回撤相比较，而回撤时期 1 的回撤比率比市场平均上证 50 指数的 18%还要小，可见 PPO 在控制回撤方面确实具有显著的优势。其他四个回撤时期的回撤比率均小于 11.5%，作为一个具有一定交易经验的正常人来说，小于 11.5%的回撤比率，是完全可以接受的。

4.3.5 回撤分析

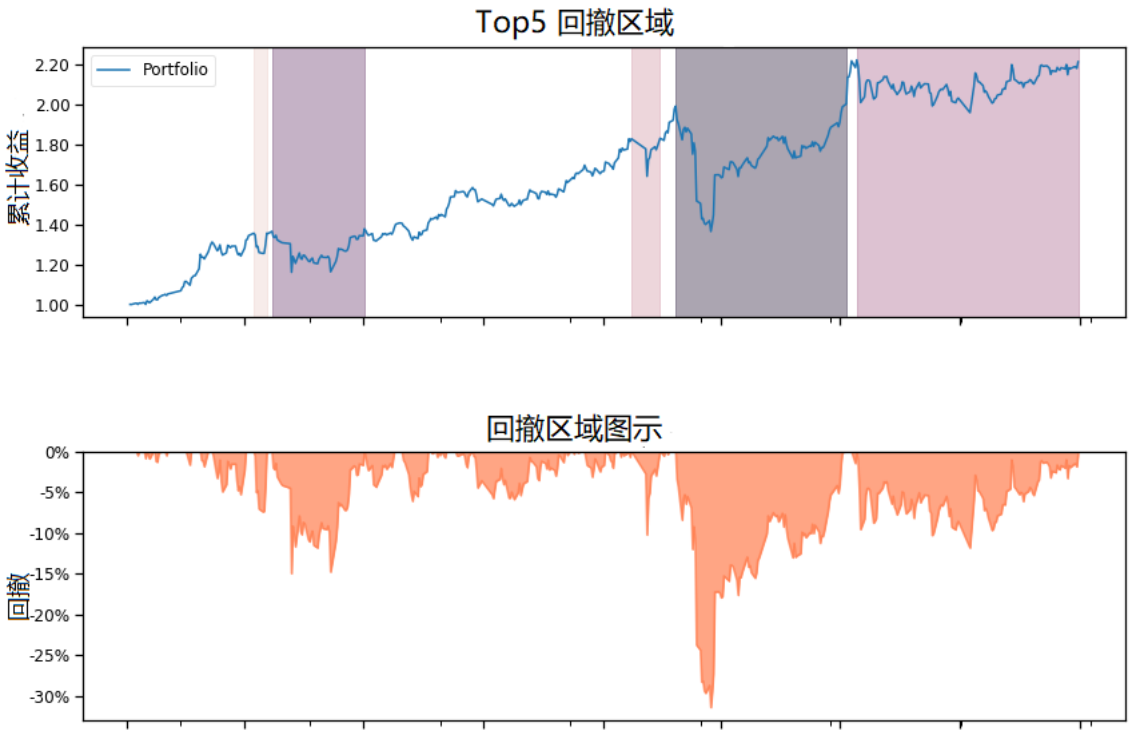

图 4-7 DDPG 最大回撤统计

图 4-7 显示的为 DDPG 智能体的回撤示意曲线，之所以选择 DDPG 智能体，是因为其收益曲线的波动比较剧烈，有着很强的代表性。如图，每幅图中的第一张图为 Top5

回撤区域，使用不同的颜色进行标出。第二张图为回撤区域内的累计收益走势。可以很直观的看出，在市场波动较为剧烈的情况下，智能体的回测也非常大，并且在 20 年 3 月市场崩盘的行情下，回撤达到了最大值。值得一提的是，智能体在每次回撤之后，累计收益又重新创了新高，这确实是一个非常了不起的成就。总的来说，强化学习智能体在控制回撤的能力上虽然不禁让人满意，但它却能做到在较大回撤后将累计收益进行修复，这是强化学习算法的优势所在。

第 5 章 总结与展望

本文使用了强化学习训练了五个股票交易的智能体，所有智能体的表现都优于上证 50 指数。从实验结果来看，强化学习在股票交易领域确实存在一定的实用性。与此同时，也不难发现，确定性策略和随机性策略的智能体有着不同的交易风格，前者比后者更为激进。我们可以考虑在牛市中使用确定性策略的智能体，在熊市中使用随机性策略的智能体。同时，如果一定要在这 5 个算法中选择最优的算法，或许 PPO 是一个不错的选择，其在管控风险的同时也能取得不错的收益。

除了 PPO 和 A2C 可以多个环境并行收集数据以外，其他算法都只能单线程运行，所以运行过程比较缓慢。运行的总耗时为 2 天整，但训练的结果却没能让人满意，预计还需要更多的训练时间。所以，在之后的训练过程中，我会尽可能采用可以多线程运行的算法。同时，如果可以的话，我会尝试对单线程的 DDPG 类算法进行改进，加入支持多线程的运行机制。

对于之后的工作，我准备去探索和使用更多有趣复杂的模型[24]，处理更大规模的数据，如沪深 300 指数成分股。我还准备给状态空间添加更多的特征，如添加交易滑点和流动性模型，结合基本面分析的指标，自然语言处理分析的市场情绪指标。这些工作将会对模型的稳定性有着很大的提升。

致谢

不知不觉中，大学生活已经进入了倒计时，非常感谢大学里帮助过我的老师以及同学。在你们的帮助下，我才能度过一个又一个的难关，个人也得到了极大的成长。同时也非常感谢我的毕设指导导师——邓磊老师和倪炜老师，给我提供选题、实验及论文书写上的帮助，给予了我很多优质的建议和耐心的指导，让整个毕业设计进行得井井有条。

感谢中南大学软件学院 1705 班的同学们。在过去的四年里，我们一起度过了许多美好的时光。同时，也感谢中南大学这么好的平台，让我能够收获知识，不断成长，变得越来越出色。

感谢我的父母。他们都给予了我无微不至的关怀。他们为我创造了求学的物质条件，同时也给予了我最大的精神支持！未来的路还很长，希望毕业不是结束，而会成为一个崭新的开始。

黄辉晴于中南大学

2021 年 6 月

参考文献

[1] Markowitzz, H. Portfolio selection[J]. The Journal of Finance, vol. 7, No. 1, pp. 77-91, 1952.

[2] Francesco Bertoluzzoa, and Marco Corazza. Testing different reinforcement learning configurations for financial trading: introduction and applications[J]. Procedia Economics and Finance, vol. 3, pp. 68-77, 2012.

[3] Volodymyr Mnih, et al. Human-level control through deep reinforcement learning[J]. Nature, pp. 529-533, 2015.

[4] M. Varela, O. Viera, and F. Robledo. A Q-Learning Approach for Investment Decisions [M]. Springer International Publishing, 2016.

[5] 孙洪永. 基于深度强化学习的量化投资策略研究[D]. 电子科技大学, 2020.

[6] Quang-Vinh Dang. Reinforcement learning in stock trading[M]. Advances in Intelligent Systems and Computing, vol 1121. Springer, Cham, 2020-01.

[7] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. Deep direct reinforcement learning for financial signal representation and trading[M]. IEEE Transactions on Neural Networks and Learning Systems, vol. 28, pp. 1–12, 2016-02.

[8] Jinke Li, Ruonan Rao, and Jun Shi. Learning to trade with deep actor critic methods[J]. 2018 11th International Symposium on Computational Intelligence and Design (ISCID), vol. 02, pp. 66–71, 2018.

[9] Yuxin Wu and Yuandong Tian. Training agent for first-person shooter game with actor-critic curriculum learning[M]. in International Conference on Learning

Representations (ICLR), 2017.

[10] Hongyang Yang, Xiao-Yang Liu, and Qingwei Wu. A practical machine learning approach for dynamic stock recommendation[J]. in IEEE TrustCom/BiDataSE, pp. 1693–1697, 2018-08.

[11] D. Silver et al. Mastering the game of Go with deep neural networks and tree search[J]. *Nature* 529(7587): 484-489, 2016.

[12] Mis 铭, 方小莉, 寒曦. 量化交易(Quantitative Trading)[EB/OL]. https://wiki.mbalib.com/wiki/量化交易, 2013-08-17.

[13] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym[M]. 2016.

[14] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines[EB/OL]. https://github.com/hill-a/stable-baselines, 2018.

[15] Volodymyr Mnih, Adria Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning[M]. The 33rd International Conference on Machine Learning, 2016-02.

[16] Timothy Lillicrap, Jonathan Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning[M]. International Conference on Learning Representations (ICLR) 2016-09.

[17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms[J]. arXiv:1707.06347, 2017-07.

[18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor[J]. arXiv:1801.01290, 2018-08-08.

[19] Scott Fujimoto, Herke van Hoof, David Meger. Addressing Function Approximation Error in Actor-Critic Methods[J]. arXiv:1802.09477, 2018-10-22.

[20] Angle Roh, 山林, 沙漠之鹰. 夏普比率(Sharpe Ratio)[EB/OL]. https://wiki.mbalib.com/wiki/夏普比率, 2010-12-28.

[21] 雨非林. Omega 比率：投资绩效的终极指标?[EB/OL]. https://www.weivol.cn/2019/01/omega-ratio/, 2019-01-22.

[22] Xiusidun, 鲈鱼. 年化收益率(annualized rate of return) [EB/OL]. https://wiki.mbalib.com/wiki/年化收益率, 2016-01-15.

[23] Xiao-Yang Liu, Hongyang Yang, Qian Chen, Runjia Zhang, Liuqing Yang, Bowen Xiao, Christina Dan Wang. FinRL: A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance[J]. arXiv:2011.09607, 2020-11-19

[24] Lu Wang, Wei Zhang, Xiaofeng He, and Hongyuan Zha. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation, in Conference on Knowledge Discovery and Data Mining (KDD)[M]. pp. 2447–2456, 2018-07.

[25] M. G. Bellemare et al. The arcade learning environment: an evaluation platform for general agents[M]. *IJCAI*, 2015.

[26] Thomas G. Fischer. Reinforcement learning in financial markets —— a survey[M] FAU Discussion Papers in Economics 2018-12.

[27] D. Pfau, and O. Vinyals. Connecting generative adversarial networks and actor-critic methods[J]. *arXiv preprint arXiv: 1610.01945*, 2016.

[28] Stelios Bekiros. Heterogeneous trading strategies with adaptive fuzzy actor-critic reinforcement learning: A behavioral approach[J]. Journal of Economic Dynamics and Control, vol. 34, pp. 1153–1170, 2010-06.

[29] Zihao Zhang. Deep reinforcement learning for trading[J]. ArXiv 2019, 2019-11.

[30] Zhuoran Xiong, Xiao-Yang Liu, Shan Zhong, Hongyang Yang, and A ·Elwalid. Practical deep reinforcement learning approach for stock trading[M]. NeurIPS Workshop on Challenges and Opportunities for AI in Financial Services: the Impact of Fairness, Explainability, Accuracy, and Privacy, 2018.

[31] Quantopian. Pyfolio: A toolkit for portfolio and risk analytics in python[EB/OL]. https://github.com/quantopian/pyfolio, 2019.

[32] HAN SHUANG LIN. 证券投资大辞典[M]. 1993.

[33] 胡奇英, 刘建庸. 马尔可夫决策过程引论[M]. 西安电子科技大学出版社, 2000.

[34] V Konda Actor-critic algorithms[J]. Siam Journal on Control & Optimization. 2003, 42(4):1143-1166.

[35] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic Policy Gradient Algorithms[M]. Proceedings of the 31st International Conference on Machine Learning (ICML-14), pages 387–395, 2014.

[36] J B Heaton, N G Polson, J H Witte. Deep learning for finance: deep portfolios[J]. Applied Stochastic Models in Business & Industry, 2017, 33(1).

[37] Kara Y, Boyacioglu M A, Baykan K. Predicting direction of stock price index movement

using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange[J]. Expert systems with Applications, 2011, 38(5):5311-5319.

[38] Tsinaslanidis P E, Kugiumtzis D. A prediction scheme using perceptually important points and dynamic time warping[J]. Expert Systems with Applications, 2014, 41(15):6848-6860.