# 第5章：
# 深度强化学习Q网络

周炜星 谢文杰

华东理工大学金融学系

2023年秋

# 纲要

# 纲要

# 智能策略

智能系统有不同类型，有不同层次，融入人类生活的方方面面：

- 计算智能
- 感知智能
- 决策智能
- 认知智能
- 通用智能

一直以来，人类在决策智能方面一直遥遥领先于机器，直到2016年AlphaGo横空出世，智能机器在围棋上打败了围棋世界冠军李世石，使智能机器的决策惊艳全球。AlphaGo用到的关键技术之一就是深度强化学习。

# 策略函数简单示例：策略函数与Q表格

策略函数表示一个映射关系，从状态空间映射到动作空间。

表 1: 基于不同天气和动作的收益矩阵

| | 下雨 | 不下雨 |
|---|---|---|
| 带伞 | 1 | −1 |
| 不带伞 | −2 | 1 |

# 策略函数简单示例

智能体最优策略是下雨带伞，不下雨就不带伞，用下表表示：

表 2: 基于不同天气最优策略

| 天气 | 最优动作 |
| --- | --- |
| 下雨 | 带伞 |
| 不下雨 | 不带伞 |

# 策略函数简单示例

如果智能体的偏好异于常人，情愿淋雨也不要打伞，对打伞的行为表现出厌恶，则基于智能体行为偏好的收益矩阵可以表示为：

表 3: 厌恶打伞的智能体的收益矩阵

| | 下雨 | 不下雨 |
|---|---|---|
| 带伞 | −2 | −2 |
| 不带伞 | 0 | 1 |

# 策略函数简单示例

厌恶打伞的智能体的最优策略为所有天气状况下都不打伞。因此，最优策略可以简单表示为矩阵形式：

表 4: 厌恶打伞的智能体的最优策略

| 天气 | 最优动作 |
|---|---|
| 下雨 | 不带伞 |
| 不下雨 | 不带伞 |

# Q表格

强化学习中的状态-动作值矩阵决定了智能体最优化策略。在日常生活中，价值矩阵或行为策略时时刻刻影响着人类的行为决策。一般而言，状态-动作值矩阵(Q-table)可以表示如下：

$$
\begin{array}{c}
 \\
s_1 \\
s_2 \\
s_3 \\
\vdots \\
s_{n-1} \\
s_n
\end{array}
\begin{array}{c}
\begin{array}{ccccccc}
a_1 & a_2 & a_3 & \cdots & a_{m-1} & a_m
\end{array} \\
\begin{pmatrix}
Q_{1,1} & Q_{1,2} & Q_{1,3} & \cdots & Q_{1,m-1} & Q_{1,m} \\
Q_{2,1} & Q_{2,2} & Q_{2,3} & \cdots & Q_{2,m-1} & Q_{2,m} \\
Q_{3,1} & Q_{3,2} & Q_{3,3} & \cdots & Q_{3,m-1} & Q_{3m} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
Q_{n-1,1} & Q_{n-1,2} & Q_{n-1,3} & \cdots & Q_{n-1,m-1} & Q_{n-1,m} \\
Q_{n,1} & Q_{n,2} & Q_{n,3} & \cdots & Q_{n,m-1} & Q_{n,m}
\end{pmatrix}
\end{array}
$$

# Q网络

深度强化学习与强化学习的最大区别在于，深度强化学习将状态-动作值矩阵用深度神经网络模型进行了替换，如图1所示：

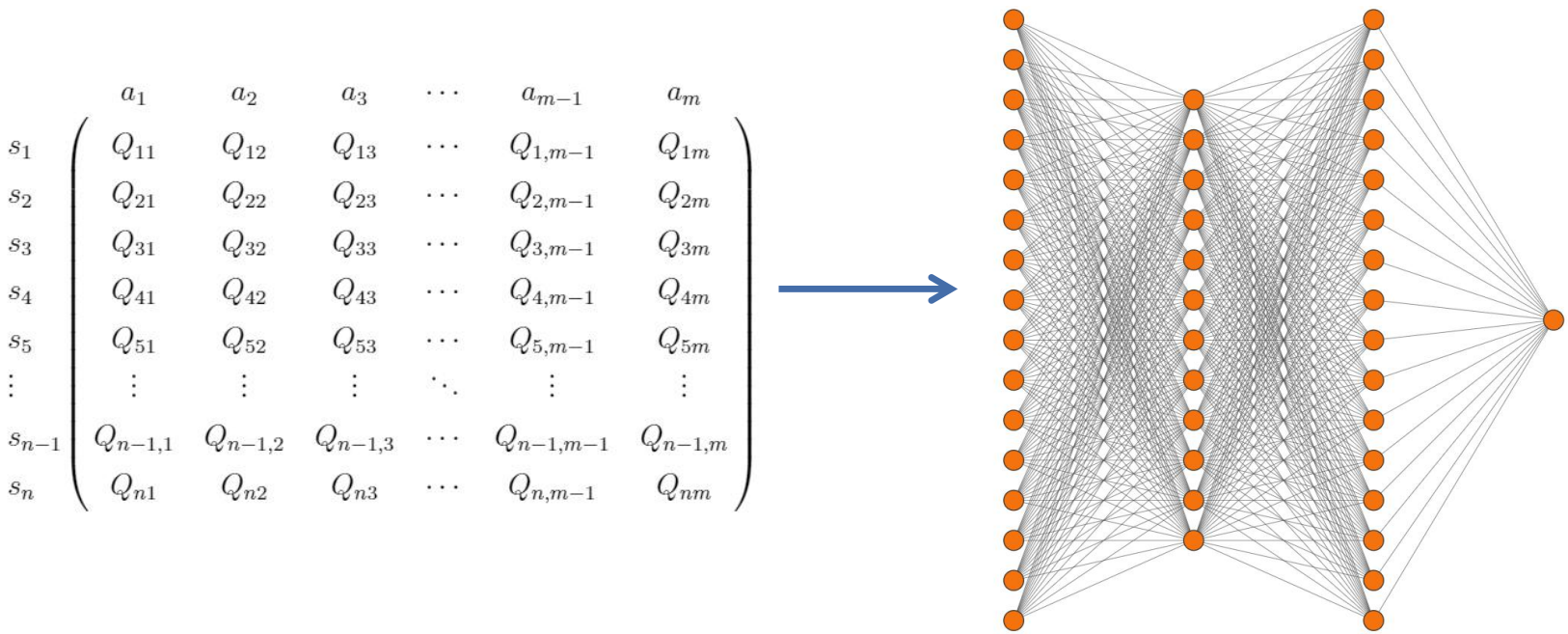



图 1: 深度强化学习用深度神经网络模型替换状态-动作值矩阵示意图

# 策略函数与Q网络

深度Q网络（Deep Q Network，DQN）通过深度神经网络拟合状态-动作值，智能体策略函数用深度Q网络表示，能够输出给定状态下每个动作的Q值，如图所示。

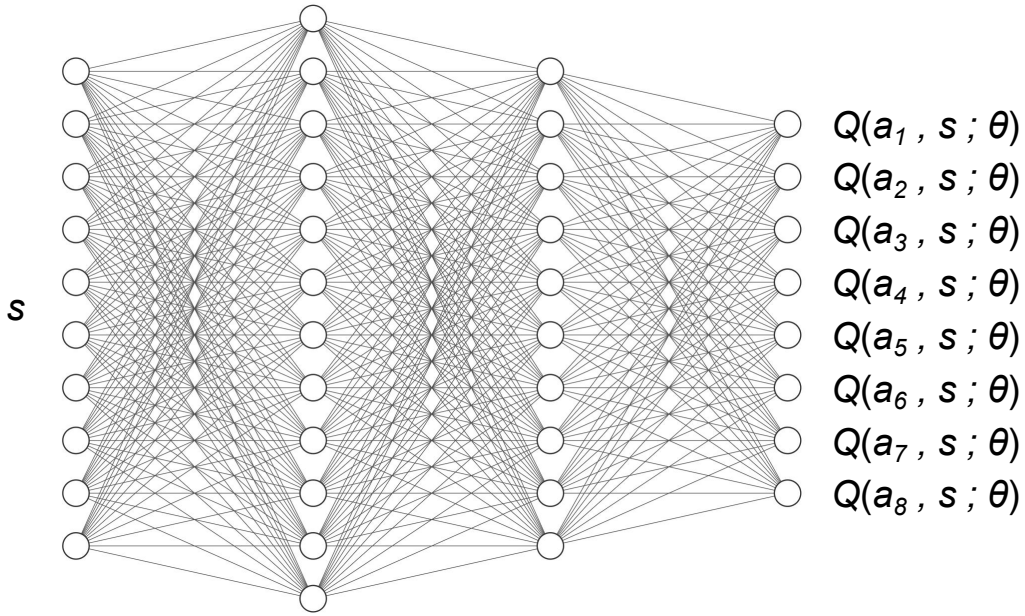



图 2: 深度Q网络（Deep Q Network）示意图

# 纲要

# Deep Q Network

DQN算法全称是Deep Q Network，基于经典强化学习算法Q-learning演化而来，Q-learning作为强化学习的重要算法，有着悠久的历史，在强化学习发展和应用过程中发挥了重要作用。在Q-learning算法中，状态-动作值函数$Q(s,a)$的更新公式为：

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left( r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right). \quad (1)$$

深度强化学习和经典强化学习的最大区别在于，深度强化学习算法中值函数或策略函数一般使用深度神经网络逼近或近似，用参数化的状态-动作值函数$Q(s,a;\theta)$逼近$Q(s,a)$，可如下表示：

$$Q(s,a) = Q(s,a;\theta). \quad (2)$$

# Deep Q Network

深度强化学习算法在更新过程中不直接对状态-动作值函数$Q(s,a;\theta)$的数值进行更新，而是更新近似状态-动作值函数$Q(s,a;\theta)$的深度神经网络模型参数$\theta$，可表示为：

$$\theta_k = \theta_{k-1} + \Delta\theta_k, \tag{3}$$

其中，$\theta_k$表示第$k$迭代时深度神经网络模型的参数，$\Delta\theta_k$表示第$k$迭代时深度神经网络模型参数的更新量或变化量。

# 经验回放（Experience Replay）

经典的DQN算法中有一个关键技术，叫经验回放。智能体在与环境交互过程中获得的经验数据会保存在经验池（Replay Buffer）之中。经验池中的数据存放形式如下：

$$\begin{aligned}
&\langle s_0, a_0, r_0, s_1\rangle,\\
&\langle s_1, a_1, r_1, s_2\rangle,\\
&\langle s_2, a_2, r_2, s_3\rangle,\\
&\vdots\\
&\langle s_n, a_n, r_n, s_n\rangle.
\end{aligned} \tag{4}$$

经验池存储满后，我们可以将旧的经验样本数据剔除，保存新的经验样本数据。

# 目标网络（Target network）

为了训练过程更加稳定，DQN算法引入了目标网络的概念。强化学习Q-learning算法的更新公式（1）迭代收敛时：

$$\left(r+\gamma \max_{a'} Q(s', a')\right) - Q(s, a) \to 0. \tag{5}$$

DQN算法的训练过程是使状态-动作值$Q(s, a)$趋近$r+\gamma \max_{a'} Q(s', a')$，即：

$$Q(s, a) \to \left(r+\gamma \max_{a'} Q(s', a')\right). \tag{6}$$

DQN算法将$r+\gamma \max_{a'} Q(s', a')$作为TD目标值，在智能体进行更新和采样过程中目标值$r+\gamma \max_{a'} Q(s', a')$是一个不断变化的。

# 目标网络（Target network）

DQN算法引入了目标网络。原始网络被称为行为网络（Behavior network），目标网络的结构与原始状态-动作价值网络$Q(s,a;\theta)$的结构相同，具有相同的深度神经网络结构，只是参数值不一样。目标网络表示为$Q(s,a;\theta^-)$，$\theta^-$为目标网络参数。目标网络和行为网络的更新频率不同。为了模型训练的稳定性，DQN算法在一定的更新步骤内保持目标网络不变，用来计算TD目标值：

$$y_t = r_t + \gamma \max_{a'} Q(s',a';\theta^-). \tag{7}$$

固定的目标更容易稳定学习过程。在DQN算法中引入目标网络的操作就如同初学者学习射飞镖，如果标靶随机移动，初学者很难掌握射中标靶的技能；反之，如果标靶固定，初学者就更容易掌握射中标靶的技能。

# 纲要

# DQN算法

DQN算法先初始化状态-动作值函数$Q(s,a;\theta)$的参数$\theta$，然后初始化目标网络，即初始化状态-动作值函数$Q(s,a;\theta^-)$的参数$\theta^- = \theta$。在与环境交互过程中，智能体采用$\epsilon$-贪心策略生成轨迹，$\epsilon$-贪心策略表示为：

$$\pi(s,a) = \begin{cases} 1-\epsilon+\frac{\epsilon}{|\mathcal{A}|}, & a = \arg\max_a Q(s,a;\theta) \\ \frac{\epsilon}{|\mathcal{A}|}, & a \neq \arg\max_a Q(s,a;\theta) \end{cases}, \tag{8}$$

其中，$|\mathcal{A}|$表示动作空间$\mathcal{A}$中动作数量。在智能体基于$\epsilon$-贪心策略选择动作时，无需目标网络$Q(s,a;\theta^-)$参与计算。

# DQN算法

DQN算法将$\epsilon$-贪心策略采样得到的经验数据四元组$\langle s, a, r, s' \rangle$存入经验池，经验池积累了一定数量的经验数据后，智能体可以从经验池中随机采样小批量的经验数据来更新行为网络，首先针对每个四元组$\langle s_i, a_i, r_i, s_i' \rangle$计算TD目标值

$$y_i = r_i + \gamma \max_{a'} Q(s_i', a'; \theta^-), \tag{9}$$

同时，TD误差表示为：

$$\delta_i = y_i - Q(s_i, a_i; \theta) = r_i + \gamma \max_{a'} Q(s_i', a'; \theta^-) - Q(s_i, a_i; \theta), \tag{10}$$

我们将小批量经验数据代入损失函数，计算TD误差的均方和：

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - Q(s_i, a_i; \theta))^2, \tag{11}$$

其中，$n$为小批量经验数据大小。

# DQN算法

损失函数$J(\theta)$的梯度为：

$$\nabla J(\theta) = -\frac{2}{n}\sum_{i=1}^{n}(y_i - Q(s_i, a_i; \theta))\nabla Q(s_i, a_i; \theta). \tag{12}$$

损失函数$J(\theta)$越小越好。我们采用梯度下降算法更新深度神经网络模型参数，行为网络参数更新公式为：

$$\theta = \theta - \alpha\nabla J(\theta). \tag{13}$$

目标网络的参数可以不用频繁更新，待行为网络更新一定步数后，我们更新目标网络参数：

$$\theta^{-} = \theta. \tag{14}$$

# DQN算法伪代码

**Algorithm 14:** DQN 算法伪代码

**Input:** 状态空间 $\mathcal{S}$，动作空间 $\mathcal{A}$，折扣系数 $\gamma$ 以及环境 Env
初始化状态-动作值函数 $Q(s,a;\boldsymbol{\theta})$ 的参数 $\boldsymbol{\theta}$
初始化目标网络 $Q(s,a;\boldsymbol{\theta}^-)$ 的参数 $\boldsymbol{\theta}^- = \boldsymbol{\theta}$
**Output:** 最优策略 $\pi^*$

**for** $k = 0, 1, 2, 3, \cdots$ **do**
  % 每次循环针对一条轨迹
  初始化状态 $s$
  **for** $t = 0, 1, 2, 3, \cdots, T$ **do**
    采用 $\epsilon$-贪心策略产生一步轨迹 $\langle s, a, r, s' \rangle$，并存入经验池
    **if** 到了需要更新参数的时候 **then**
      从经验池中随机采样小批量 $n$ 个状态转移序列对 $\langle s, a, r, s' \rangle$，针对每个序列 $i$ 计算 TD 目标值
      **if** $s'$ 是终止状态 **then**
        $y_i = r_i$
      **else**
        $y_i = r_i + \gamma \max_{a'} Q(s', a'; \boldsymbol{\theta}^-)$
      计算小批量 $n$ 个状态转换序列的损失函数 $J(\boldsymbol{\theta})$ 及其梯度：

$$\nabla J(\boldsymbol{\theta}) = -\frac{2}{n}\sum_{i=1}^{n}(y_i - Q(s_i, a_i; \boldsymbol{\theta}))\nabla Q(s_i, a_i; \boldsymbol{\theta}) \tag{5.15}$$

      更新网络参数 $\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \nabla J(\boldsymbol{\theta})$
    % 参数 $\boldsymbol{\theta}$ 更新 $C$ 次后更新一次 $\boldsymbol{\theta}^-$
    **if** 间隔 $C$ 步 **then**
      $\boldsymbol{\theta}^- = \boldsymbol{\theta}$
返回最优参数 $\boldsymbol{\theta}$，并得到最优策略 $\pi^*$
**for** $s \in S$ **do**
  $\pi^*(s) = \arg\max_a Q(s, a; \boldsymbol{\theta})$

# 纲要

# Double DQN背景

经典Q-learning算法和DQN算法在估计状态-动作值函数$Q(s,a)$时都存在过估计问题，即得到的状态-动作值函数Q值高于真实状态-动作值。过估计问题的原因有很多，主要是由值函数迭代逼近或优化过程中的最大化操作$\max_a Q(s,a)$引入。经典DQN中的TD目标计算公式为：

$$y_t = r_t + \gamma \max_{a'} Q(s', a'; \theta^-). \tag{15}$$

将TD目标计算公式等价变换后，可重写为：

$$y_t = r_t + \gamma Q(s', \arg\max_{a'} Q(s', a'; \theta^-); \theta^-). \tag{16}$$

# Double DQN背景

因此，DQN算法中的TD目标计算可以分解成两步来进行，第一步为选择最优动作：

$$a^* = \arg\max_{a'} Q(s', a'; \theta^-). \tag{17}$$

第二步为计算最优动作$a^*$对应的TD目标值：

$$y_t = r_t + \gamma Q(s', a^*; \theta^-). \tag{18}$$

在计算DQN算法中TD目标值的过程中，动作选择和动作价值评估都选择了目标网络$Q(s', a'; \theta^-)$。为了克服过估计问题，双Q网络（Double DQN）算法将动作选择和动作评估做了改进。

# 双Q网络结构

双Q网络的思想简单易懂，在智能体训练中计算TD目标值时，分离动作选择过程和动作评估过程，使用不同网络进行函数近似。经典DQN算法中已经构建了两个网络，即主网络（行为网络）和目标网络。Double DQN算法使用主网络（行为网络）选择动作：

$$a^* = \arg\max_{a'} Q(s', a'; \theta). \tag{19}$$

同时，Double DQN算法用目标网络进行动作评估，计算最优动作对应的TD目标值：

$$y_t = r_t + \gamma Q(s', a^*; \theta^-). \tag{20}$$

融合动作选择和动作评估过程，我们可以得到TD目标值为：

$$y_t = r_t + \gamma Q(s', \arg\max_{a'} Q(s', a'; \theta); \theta^-). \tag{21}$$

# 双Q网络结构

同样，我们可以得到TD误差为：

$$\delta_t = r_t + \gamma Q(s', \arg\max_{a'} Q(s', a'; \theta); \theta^-) - Q(s_i, a_i; \theta). \quad (22)$$

Double DQN算法基于TD误差进行模型训练，参数更新公式可表示为：

$$\theta = \theta + \alpha \left( r_t + \gamma Q(s', \arg\max_{a'} Q(s', a'; \theta); \theta^-) - Q(s_i, a_i; \theta) \right) \nabla Q(s_i, a_i; \theta) \quad (23)$$

上式中的梯度$\nabla Q(s_i, a_i; \theta)$为$Q(s_i, a_i; \theta)$增加的方向，当公式中

$$r_t + \gamma Q(s', \arg\max_{a'} Q(s', a'; \theta); \theta^-) > Q(s_i, a_i; \theta), \quad (24)$$

# 双Q网络结构

我们最小化损失函数，$Q(s_i, a_i; \theta)$必须越来越逼近$r_t + \gamma Q(s', \arg\max_{a'} Q(s', a'; \theta); \theta^-)$，则参数$\theta$朝着$Q(s_i, a_i; \theta)$增加的方向更新。我们增加$Q(s_i, a_i; \theta)$，缩小$Q(s_i, a_i; \theta)$与目标值$y_t$之间的差距。当公式中

$$r_t + \gamma Q(s', \arg\max_{a'} Q(s', a'; \theta); \theta^-) < Q(s_i, a_i; \theta), \tag{25}$$

参数$\theta$朝着$Q(s_i, a_i; \theta)$减小的方向更新。我们减小$Q(s_i, a_i; \theta)$，缩小$Q(s_i, a_i; \theta)$与目标值$y_t$的差距。

简而言之，Double DQN算法的关键之处在于融合动作选择和动作评估过程：

$$Q(s', \arg\max_{a'} Q(s', a'; \theta); \theta^-). \tag{26}$$

我们可以简单分析Double DQN算法中双网络$\theta^-$和DQN算法中目标网络$\theta^-$之间的异同。

# 纲要

# Dueling DQN算法框架简介

深度强化学习算法改进的主要方向是提高算法绩效。DQN算法改进涉及很多方面，如特征提取、值函数估计、策略输出等方面都能够改进算法绩效。 Dueling DQN算法对网络结构进行了改进，提升了算法性能，Dueling DQN网络结构如图3所示。

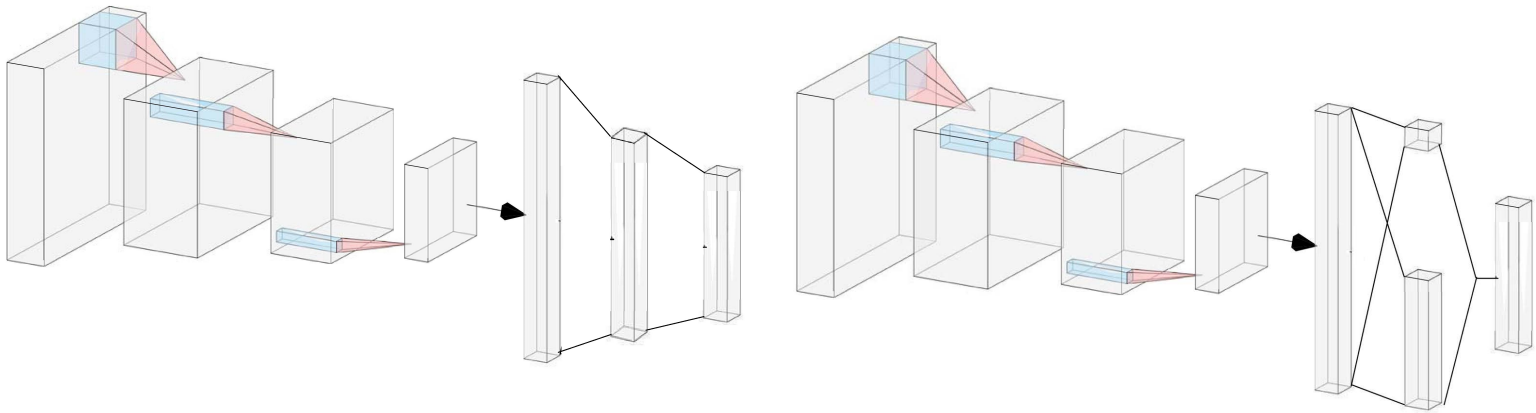


图 3: DQN（左）和Dueling DQN（右）网络结构示意图

# Dueling DQN算法核心思想

Dueling DQN核心思想是将状态-动作值函数$Q(s,a;\theta)$分解成两部分，一部分是状态值函数$V(s;\theta,\beta)$，另一部分是在状态值函数基础上选择动作$a$的优势函数$A(s,a;\theta,\alpha)$，用公式表示如下：

$$Q(s,a;\theta,\alpha,\beta)=V(s;\theta,\beta)+A(s,a;\theta,\alpha). \tag{27}$$

状态值函数$V(s;\theta,\beta)$和优势函数$A(s,a;\theta,\alpha)$都可以用深度神经网络模型表示，且能够共用部分深度神经网络结构，对应图3中Dueling DQN网络中左边部分结构，对应的模型参数为公共参数$\theta$。

# Dueling DQN算法核心思想

分离状态值函数$V(s;\theta,\beta)$和优势函数$A(s,a;\theta,\alpha)$具有一定的合理性，优势函数$A(s,a;\theta,\alpha)$可以表示为：

$$A(s,a;\theta,\alpha)=Q(s,a;\theta,\alpha,\beta)-V(s;\theta,\beta), \tag{28}$$

其中，状态值函数$V(s)$是状态-动作值函数$Q(s,a)$的加权平均值，优势函数$A(s,a;\theta,\alpha)$可以看作是动作$a$的价值$Q(s,a)$相较于平均价值$V(s)$的优势值。

# Distributional DQN

Distributional DQN算法的主要思想是，智能体在学习过程中学到的不仅仅只是状态-动作价值的一个数值，或者叫做平均值，而是状态-动作价值的分布情况。

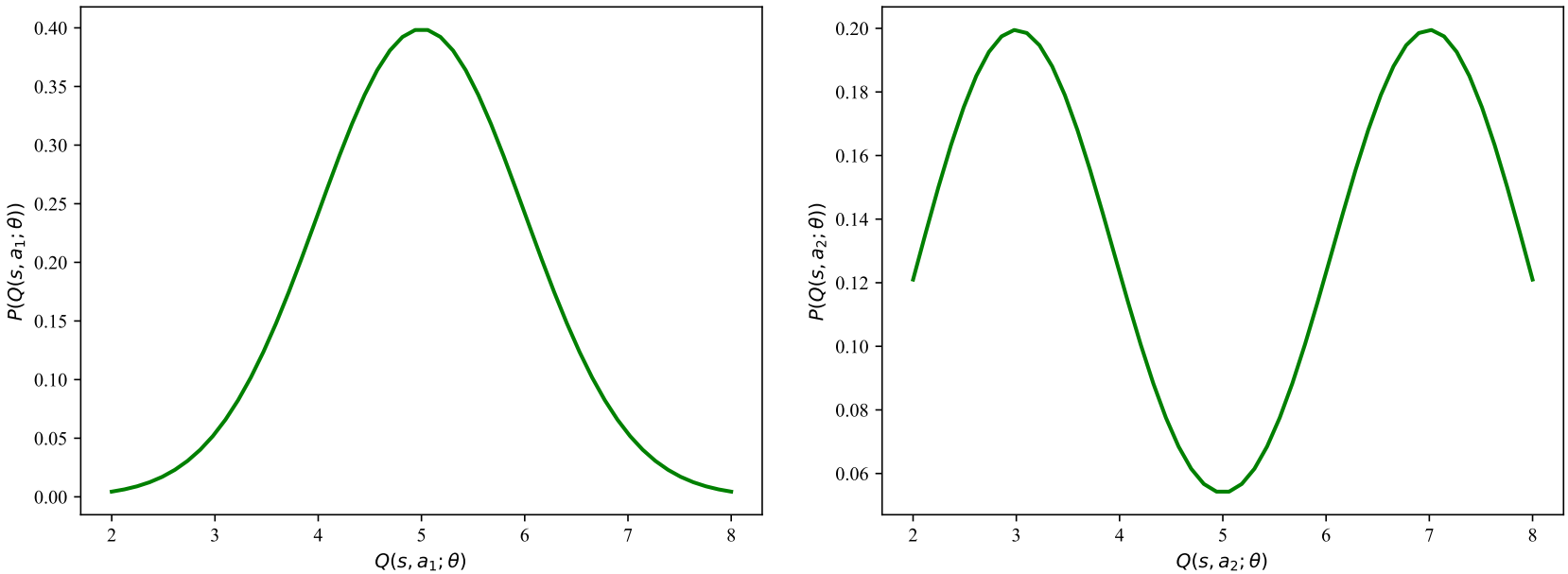



图 4: Distributional DQN算法状态-动作值$Q(s,a;\theta)$的概率分布示意图

# 纲要

# 优先级经验回放（Prioritized Experience Replay）

基于优先级经验回放的DQN算法的核心是，从经验池随机抽样经验样本时，并不是完全等概率随机抽样所有样本，而是考虑优先级的随机抽样，不同样本被抽取的概率不一样。回顾一下DQN算法参数更新公式：

$$\theta = \theta + \alpha \left( r_i + \gamma Q(s_i', \arg\max_{a'} Q(s_i', a'; \theta); \theta^-) - Q(s_i, a_i; \theta) \right) \nabla Q(s_i, a_i; \theta \quad (29)$$

将公式中的TD误差用$\delta$表示：

$$\delta_i = \left( r_i + \gamma Q(s_i', \arg\max_{a'} Q(s_i', a'; \theta); \theta^-) - Q(s_i, a_i; \theta) \right) \quad (30)$$

# 优先级经验回放（Prioritized Experience Replay）

在DQN算法更新过程中，参数$\theta$更新的幅度不仅和学习率$\alpha$有关，也与$\delta$的大小相关。DQN算法的参数更新公式可重写为：

$$\theta = \theta + \alpha\delta_i \nabla Q(s_i, a_i; \theta), \tag{31}$$

其中，梯度$\nabla Q(s_i, a_i; \theta)$决定了参数更新的方向，学习率$\alpha$和$\delta$决定了参数更新的大小。

# 基于TD误差绝对量的样本优先级

为了快速学习到最优参数$\theta$，可以偏向于对TD误差$\delta$较大的样本进行采样，即给与TD误差$\delta$较大的样本更高的优先级$p$。状态转移样本$i$的优先级可以表示为：

$$p(i) \propto |\delta_i| + \epsilon, \tag{32}$$

其中，$\epsilon$为一个极小的正数，使得每一个样本都有可能被采样。从上式可见，样本$i$的优先级正比于TD误差的绝对量$|\delta_i|$。

# 基于TD误差排序的样本优先级

我们在计算样本优先级时，将样本按照TD误差$|\delta|$从大到小排序，假设样本$i$的排序为$\mathrm{rank(i)}$，则状态转移样本$i$的优先级可以表示为：

$$p(i) = \frac{1}{\mathrm{rank(i)}}. \tag{33}$$

相较于基于TD误差绝对量的样本优先级，基于排序$|\delta_i|$的样本优先级更加稳定，因为个别样本的TD误差绝对量$|\delta_i|$变化对其它样本影响较小。在实际应用中，基于排序的样本优先级对样本异常值不敏感，样本优先级具有稳健性。

# 噪声网络DQN

在噪声网络DQN算法中，Noise网络对应状态-动作值函数$Q(s,a;\theta)$网络，将噪声加入参数$\theta$中：

$$\theta = \theta + \epsilon_\theta. \tag{34}$$

在噪声网络DQN算法中，深度神经网络参数$\theta$在加入随机噪声后，在一轮训练周期中不再加入随机噪声，以保证在一轮训练中智能体策略函数是相同的。

# 多步（Multi-step）DQN

为了得到更加准确的状态-动作值函数估计，可以采用多步（Multi-step）DQN算法，得到$n$步动作的累积回报，以及第$n+1$步的状态$s_{n+1}$，得到$n$步TD目标值：

$$y_t = \sum_{t'=t}^{t'=t+n-1} \gamma^{t'-t} r_{t'} + \gamma^n \max_{a_{n+1}} Q(s_{n+1}, a_{n+1}; \theta). \tag{35}$$

TD误差$\delta$可以表示为：

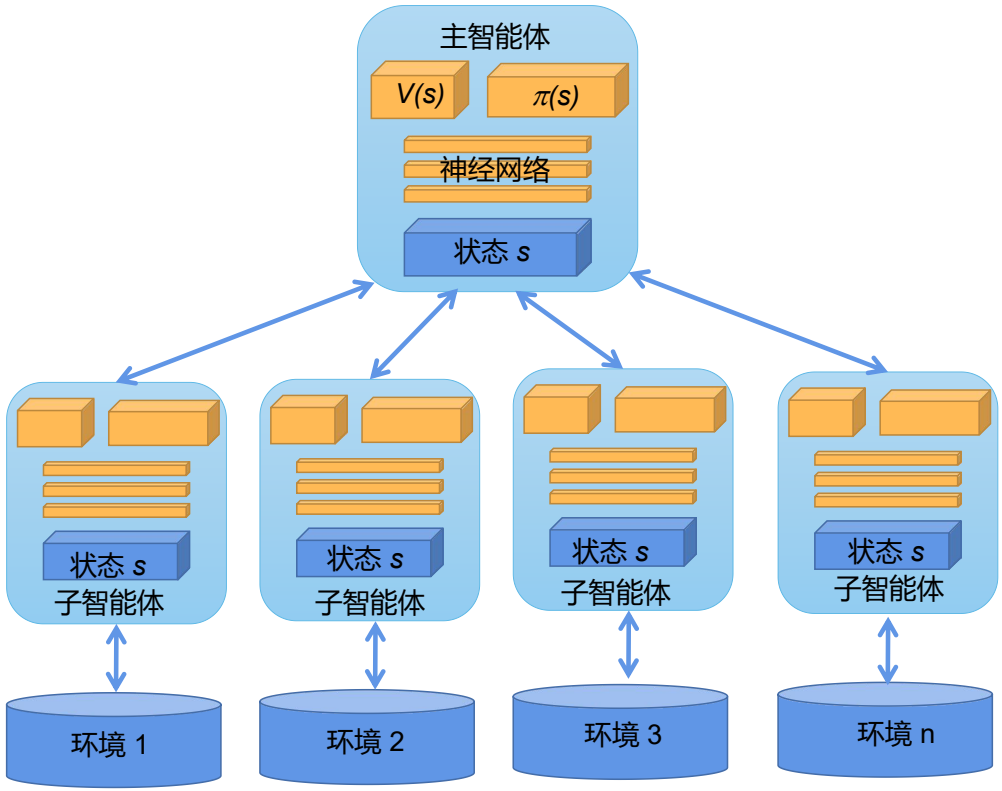
$$\delta = \left( \sum_{t'=t}^{t'=t+n-1} \gamma^{t'-t} r_{t'} \right) + \gamma^n \max_{a_{n+1}} Q(s_{n+1}, a_{n+1}; \theta) - Q(s, a; \theta). \tag{36}$$

将$n$步TD误差$\delta$代入参数更新公式，可得：

$$\theta \leftarrow \theta + \alpha \delta \nabla Q(s, a; \theta). \tag{37}$$

# 分布式训练

A3C算法的核心是，多个环境中的多个智能体对深度神经网络参数进行异步更新，其算法架构如图所示。



# DQN算法改进汇总

DQN算法改进汇总如下：

- 智能体探索能力方面
- 经验样本使用效率方面
- 梯度更新方面
- 算法原理改进方面

近年来，DQN算法的发展也取得了较大进步，特别是在工程应用中取得了举世瞩目的成果，如DeepMind的 Alpha系列智能程序在棋类、游戏、生物等领域的应用。

# 纲要

# 智能投资决策系统

基于深度强化学习的智能投资决策系统结构如图所示。

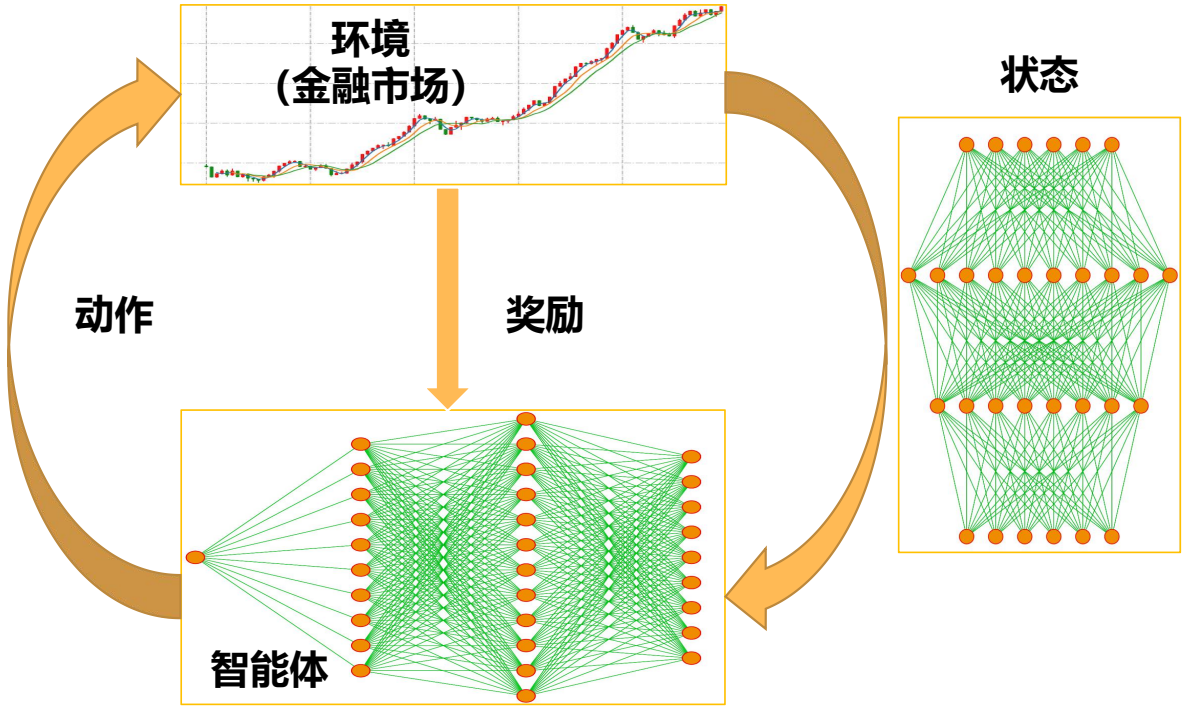
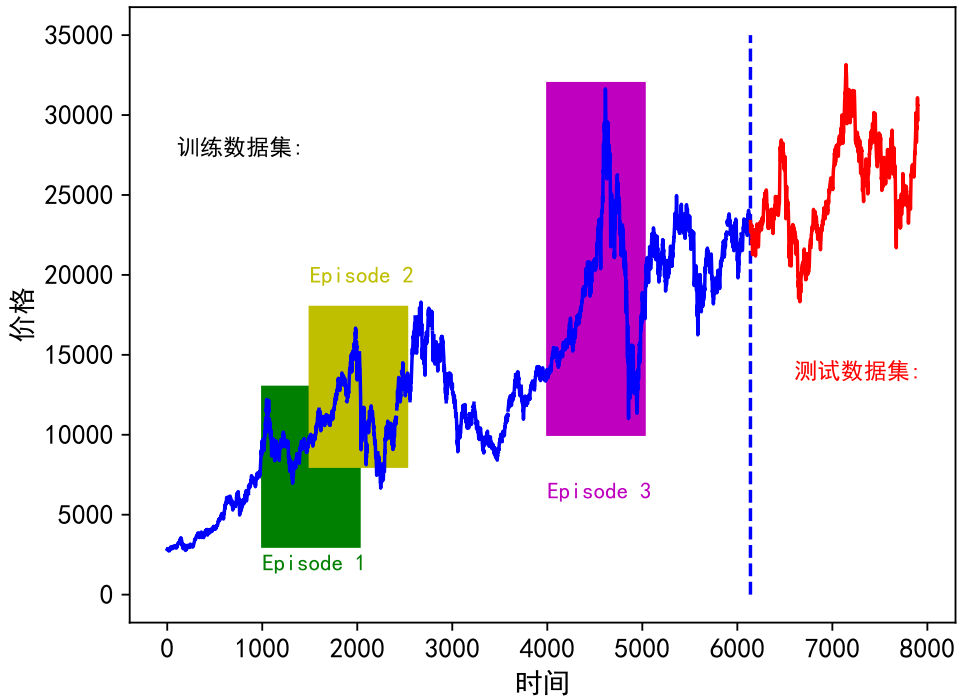



图 6: 基于深度强化学习的智能投资决策系统结构示意图

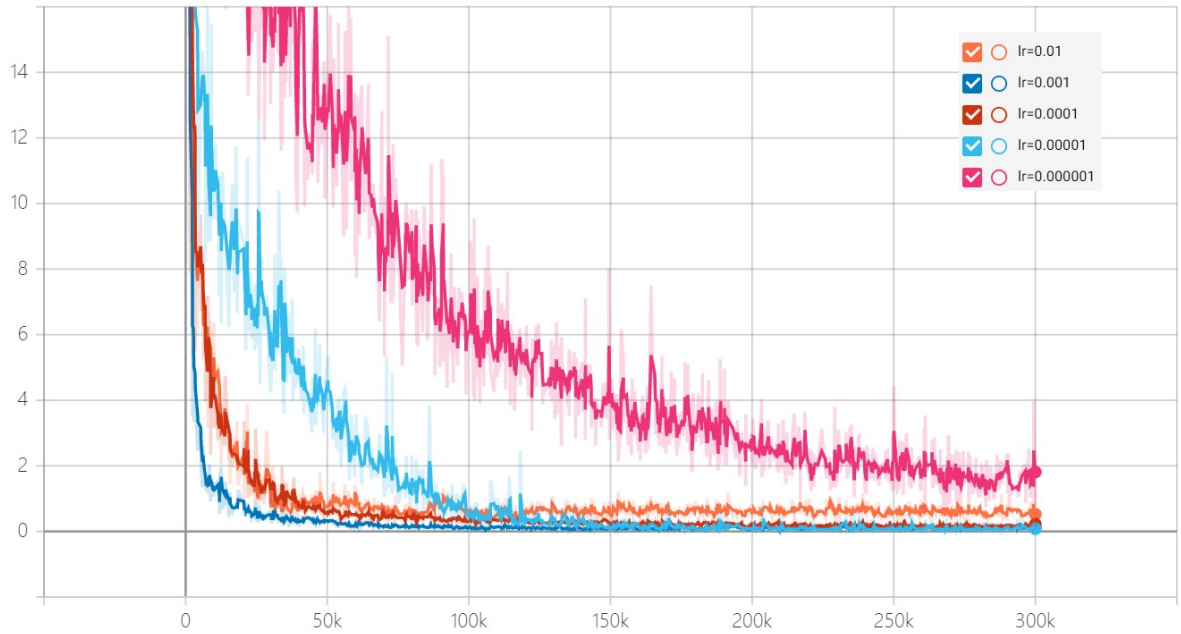
# 金融交易环境模型

我们为了保证训练数据和测试数据严格的分离，对时间序列进行了划分。2010年01月01日至2015年12月31日的价格时间序列为训练集；2016年01月01日至2016年03月31日价格时间序列为测试集。



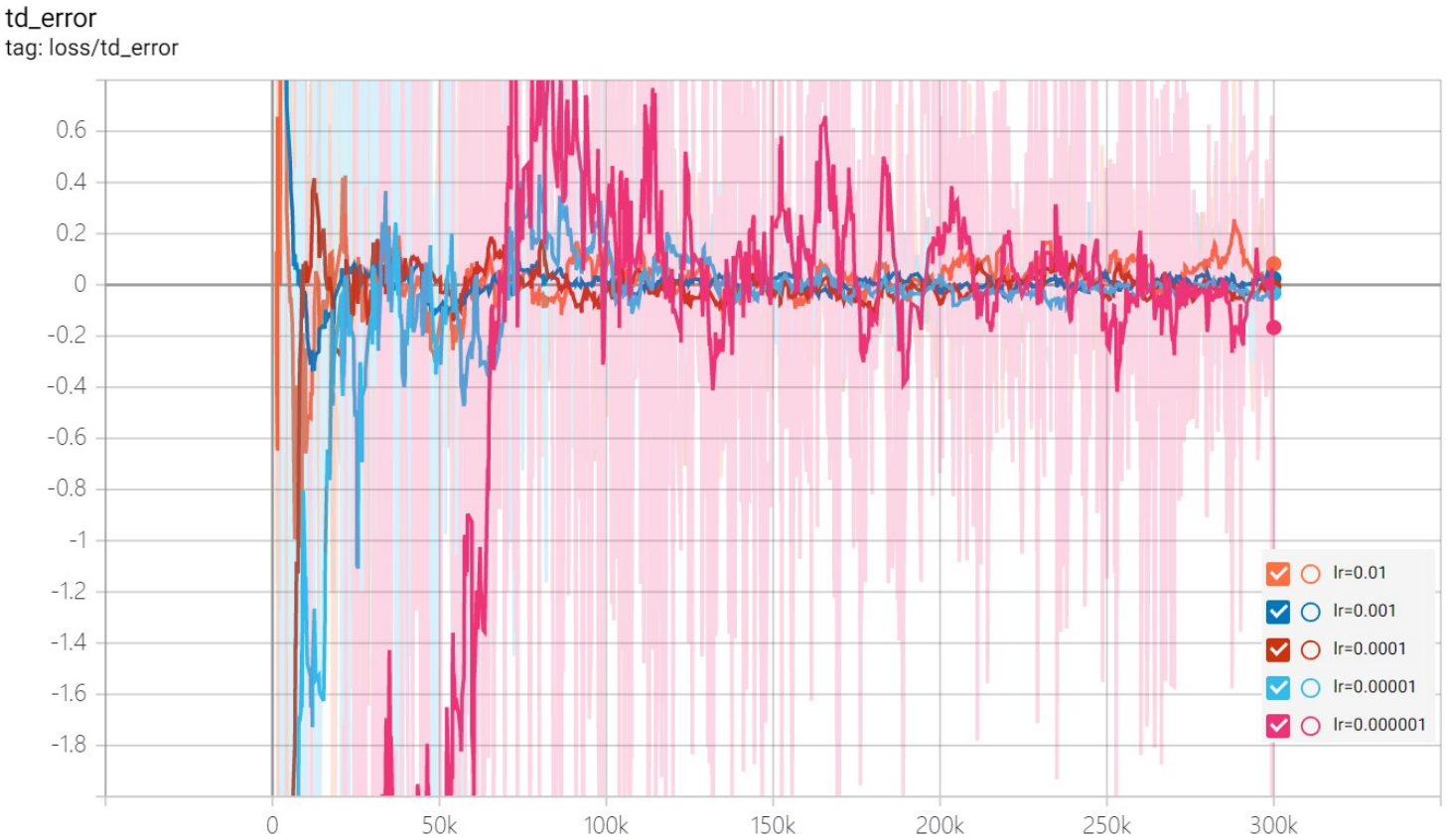
# 模型训练

Tensorboard与TensorFlow记录和分析模型训练过程中参数变化情况。Tensorboard记录的日志数据对模型的训练、程序的调整、参数的调优都很有帮助。图8给出模型在30万次迭代训练过程中损失函数变化曲线。



# 模型训练

图8显示，在五种学习率learning_rate情况下，损失函数的衰减规律类似。



# 模型训练

智能体在单个训练周期内的累积收益情况能够衡量投资策略的盈利能力：

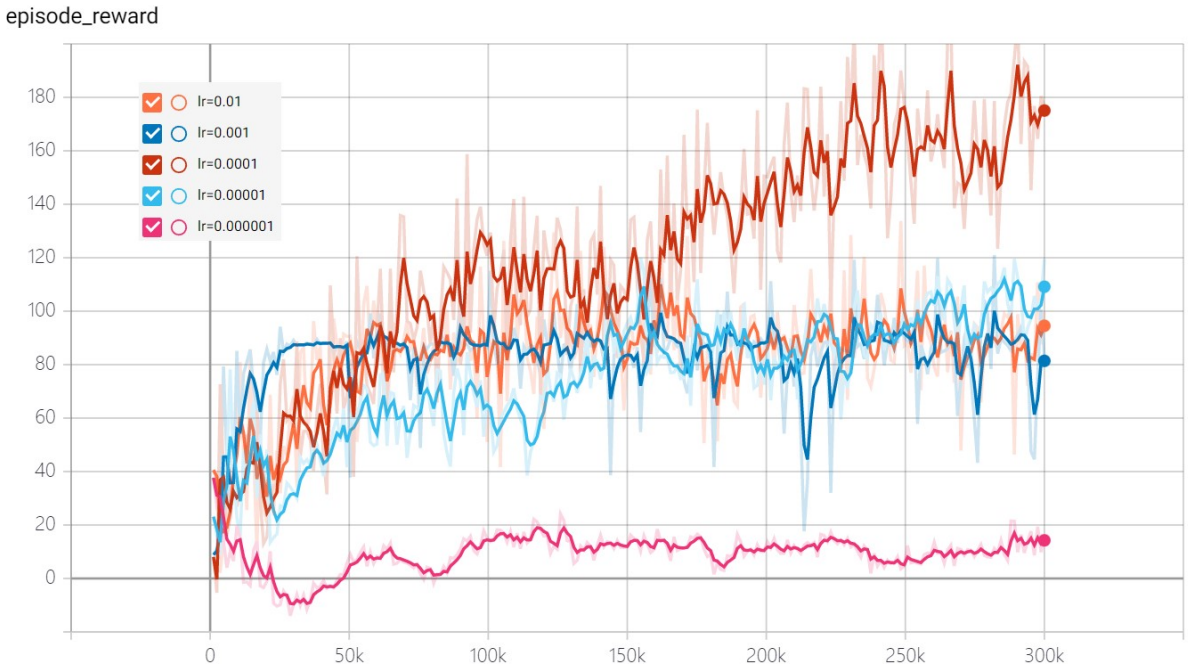



图 10: 模型训练过程中智能体单个训练周期内累积收益变化曲线

# 模型测试

在我们的示例中，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。模型测试结果如图所示。

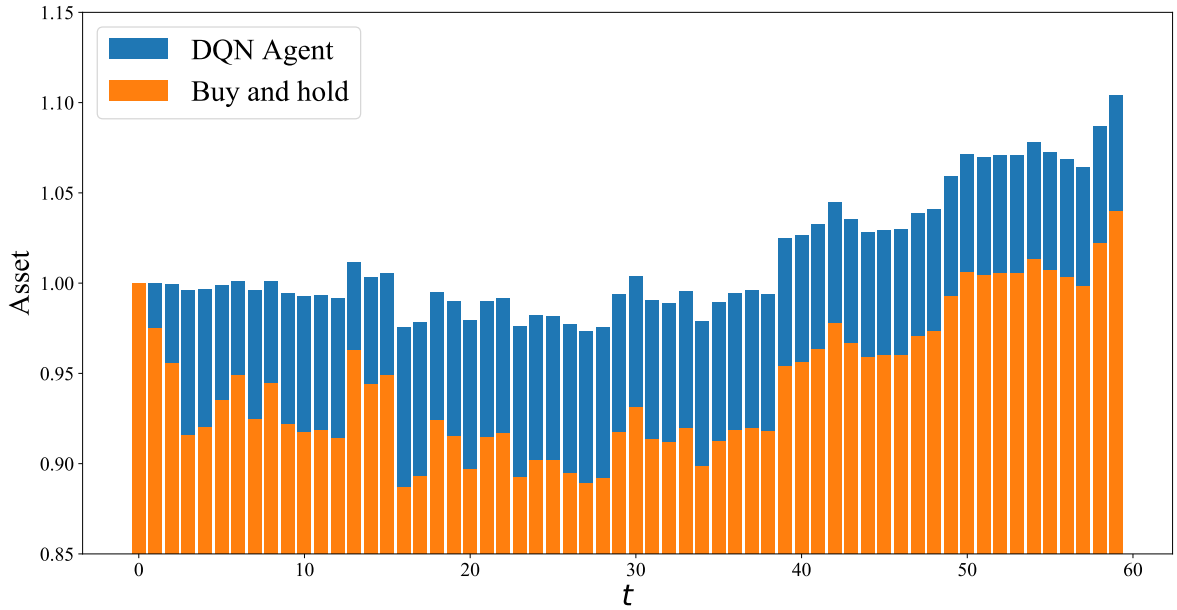



图 11: 模型测试结果

# 掌握的问题

1. DQN算法有哪些局限性？
2. DQN算法与Q-learning算法之间的区别和联系？
3. 简述DQN算法中经验回放机制？
4. DQN算法中目标网络的用途是什么？
5. DQN算法中$\epsilon$-贪心策略的用途是什么？
6. DQN算法有哪些改进算法？
7. 运用DQN实现一个智能交易强化学习系统。