分类号_____ 编 号_____

U D C_____ 密 级_____

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

本科生毕业设计（论文）

题　　目：基于强化学习的自动化函数近似

姓　　名：潘和伟

学　　号：12010112

院　　系：计算机科学与工程系

专　　业：计算机科学与技术

指导教师：史玉回 讲席教授

2024 年 6 月 7 日

诚信承诺书

1.本人郑重承诺所呈交的毕业设计（论文），是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料均真实可靠。

2.除文中已经注明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的作品或成果。对本论文的研究作出重要贡献的个人和集体，均已在文中以明确的方式标明。

3.本人承诺在毕业论文（设计）选题和研究内容过程中没有抄袭他人研究成果和伪造相关数据等行为。

4.在毕业论文（设计）中对侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

作者签名：潘和伟

2024 年 6 月 7 日

基于强化学习的自动化函数近似

潘和伟

（计算机科学与工程系　指导教师：史玉回）

[摘要]：本篇论文提出了一种基于强化学习的自动化函数近似方法。在研究中，我们创新性地将函数近似过程视为一个马尔科夫决策过程（Markov Decision Process，MDP）。在这个过程中，我们巧妙地利用 Operator Tree 来表示函数，以此作为状态；对函数的操作则作为动作；每一步动作后的得分视为奖励。通过这种方式，我们成功地将复杂的函数近似问题转化为一个决策问题，为强化学习的应用提供了可能。进一步地，我们采用了经典的深度强化学习算法：深度 Q 网络（Deep Q-Network，DQN）算法，并为其制定了一个合适的奖励函数，用于指导模型的探索、学习和优化过程。通过不断地探索和试错，模型能够自动地学习和优化函数近似的策略，从而得到了更加精准的近似结果。这种方法的提出，不仅为解决函数近似问题提供了新的思路，也为强化学习在更广泛领域的应用提供了有益的参考。

[关键词]：强化学习；深度 Q 网络；Operator Tree；函数近似

[ABSTRACT]: This paper proposes an automated function approximation method based on reinforcement learning. In our research, we innovatively treat the function approximation process as a markov decision process. Within this framework, we cleverly represent the function as a state using an operator tree, and the operations on the function serve as actions. The score after each action is viewed as a reward. By doing so, we successfully transform the complex function approximation problem into a decision-making problem, paving the way for the application of reinforcement learning. Furthermore, we adopt the classical deep reinforcement learning algorithm: Deep Q-Network algorithm, and formulate an appropriate reward function for it to guide the exploration, learning and optimization process of the model. Through continuous exploration and trial-and-error, the model can automatically learn and optimize the function approximation strategy, resulting in more accurate approximation results. The introduction of this method not only provides a new perspective for solving function approximation problems, but also offers valuable reference for the application of deep reinforcement learning in broader fields.

[Keywords]: Reinforcement Learning; Deep Q-Network; Operator Tree; Function Approximation

目录

1. 绪论

1.1 研究背景与意义

1.1.1 研究背景

在物理模拟与工程计算的场景中，所遇到的目标函数往往可能是一个复杂的组合函数，这其中可能含有三角函数、对数函数等。例如，在分析电路相关的问题中，电流、电压或功率与电阻、电容或电感等电路元件之间的关系往往会用对数函数描述；在机械振动分析中，物体的位移与时间之间的关系可能是类似正弦或者余弦的。

虽然这些复杂的函数能够准确地描述这些物理现象，但是在实际处理这些函数时，因为三角函数、对数函数等函数本身的性质：计算复杂度高或函数非凸等，对数值分析、最优化分析等问题的求解上带来不小的阻碍。

为了简化计算和提高效率，将目标函数中的三角函数和对数函数部分近似成幂函数的多项式形式是一个有效手段。通过这种近似手段，目标函数从一个含有三角函数或对数函数的复杂组合函数近似成了仅由幂函数的多项式组成的函数。这样的近似在局部函数数值分析和最优化分析上有着几个显著的优势。首先，计算复杂度降低，无需进行复杂的三角函数或对数运算，提高了运算速度。其次，近似后的函数具有良好的连续性和可导性，这对于数值分析和最优化分析有着重要作用，因为可以更容易地进行微分与积分的运算，进而支持更高效的算法设计和实现。

1.1.2 研究意义

对一个复杂的函数进行近似操作，这需要相关人员有着专业的数学知识、经验与能力。手动进行近似，这不仅费时费力、容易出错，当目标函数稍微变化，又需要重新计算，再一次增加了出错的风险。

本文希望通过强化学习算法训练一个模型能够自动化的实现如何近似一个复杂函数。相较于手动，这样不仅减少了近似过程中出错的风险、提高了效率与精准度，还对相关人员的数学要求也大幅降低。除此之外，通过显式给出模型每一步是如何处理函数的，这样还能指导研究人员如何更合理的去处理复杂的函数。

1.2 国内外研究现状

强化学习[1]来源于控制领域的一个分支。1956 年，Bellman 提出了动态规划方法[2]，这为强化学习的发展奠定了基础。在 1989 年，Watkins 提出了 Q 学习算法[3]，其结合了时间差分[4]与最优控制思想，用表格（Q 表）存储动作价值，通过迭代更新 Q 表，找到每个状态下的最有动作，最终得到最优化的策略。但是这种算法只能处理简单离散的环境。2015 年，Google DeepMind 提出了深度 Q 学习算法[5]，这是深度强化学习[6]的里程碑事件，其将深度学习[7]与强化学习相结合，使用神经网络建立状态到动作的映射，解决了无法处理复杂的连续状态的问题，还在雅达利系列的多款游戏中打破人类创下的最高记录。在此基础上，深度强化学凭借深度学习强大的表征能力，在多个领域都有了显著的成果。2018 年的 DeepCube 在解决关于魔方的组合最优化问题上取得了突破[8]。除了单智能体的场景，多智能体场景下深度强化学习算法仍有不错的发挥，例如：ROMAX[9]。在处理数学语言领域，深度强化学习实现了数学应用题自动求解器[10]，这对深度强化学习在数学语言处理领域带来了积极作用。

1.3 论文主要研究内容

近年来，深度强化学习算法在工业自动化、游戏 AI 等领域取得了令人瞩目的成果，但是深度强化学习在处理数学语言方面的研究还比较少。随着自然语言处理领域的成果：大语言模型[11]，在近几年的高速发展，数学语言处理也备受关注，不少研究将数学公式的表示方法用在大语言模型上，这提升了大语言模型在处理数学问题的能力。

本文通过将数学函数近似的过程描述成一个马尔科夫决策过程，其中状态即是函数的表示、动作是对函数的操作、合理定制奖励函数，并且成功的将大语言模型中对数学公式的表示方法与深度强化学习算法，即 OPT 与 DQN，相结合来解决数学函数近似的问题。

本文的相关工作不仅为强化学习的在数学语言处理应用领域提供了新的经验，还为解决数学问题提供了新的方向与思路。

1.4 论文结构与章节安排

本篇论文被分为 5 个章节，其中每一个章节的安排具体如下：

第 1 章绪论，主要介绍了本文的研究背景与研究意义。本章详细的介绍了强化学习的国内外研究现状，进而对本课题的研究内容和技术路线进行了分析。

第 2 章相关理论基础，对本文涉及到的概念和理论进行解释。本章阐述了强化学习的基本概念，并描述了深度强化学习相较于传统强化学习所带来的优势，着重介绍了经典的深度强化学习算法 DQN，最后介绍了常用的数学公式表示方法中 OPT 的优势。

第 3 章方法设计与实现，主要介绍了本文中所涉及到的方法的具体实现。本章详细讲解了本次研究中如何实现 DQN 算法框架及其状态空间、动作空间和奖励函数。其中状态空间在本文中指的是如何将数学公式通过 OPT 表示。

第 4 章实验结果分析，将第三章中实现的 DQN 算法进行仿真实验，并且对结果进行分析，验证实验能够实现本文的研究目的。

第 5 章总结与展望。本章节重新回顾整篇论文，总结所做的工作，分析不足的地方，最后对未来的研究工作做出规划与展望。

2. 相关理论基础

2.1 强化学习

2.1.1 基本概念

强化学习是机器学习中的一种范式和方法论。与监督学习不同，强化学习不需要预先给定的标签或数据，而是通过智能体与环境的交互来获取学习信息[12]。在强化学习中有以下几个基本概念：

（1）智能体（Agent）： 进行决策的实体，通过与环境之间的交互进行学习。

（2）环境（Environment）：描述了智能体需要完成任务的场景，其会对智能体的决策产生影响。

（3）状态（State）：当前环境的状态，为智能体提供所需的所有信息，因此状态是智能体进行决策的主要依据。

（4）动作（Action）：智能体在特定状态下可以执行的操作。动作会受到环境的影响，通常每个状态下的动作集合是有限的。

（5）奖励（Reward）：环境对智能体采取的动作给出的即时反馈，它是一个数值信号（正或负的），用来评估评估特定状态下特定动作的好坏程度。

（6）策略（Policy）：智能体在特点状态下的行为方式。策略是从状态到动作的映射。智能体与环境不断交互学习的目的就在于优化其策略，让长期累计奖励最大化。

（7）价值函数（Value Function）：评估在不同状态下，一个智能体执行某种策略可以获得的长期回报。价值函数对智能体选择动作和学习策略有着重要影响。

在强化学习中，智能体会根据当前状态按照指定策略选择合适的动作，然后观察环境的反馈（奖励），通过不断地“试错”和“学习”来更新策略，最终来找到最优的策略，让智能体能够获得最大化的长期累积奖励。

强化学习在许多领域都有应用，如工业自动化、金融贸易、游戏 AI 等。在这些应用中，强化学习可以帮助智能体学习如何执行各种任务，并通过不断地学习和优化来提高其性能。

2.1.2 马尔可夫决策过程

马尔可夫决策过程是一种序贯决策（Sequential Decision）问题的数学模型，它基于马尔可夫过程理论，用于描述智能体在具有马尔可夫性质的环境中如何做出决策以最大化长期累积的奖励[13]。MDP 提供了一个明确的理论框架来对强化学习问题进行建模。一个 MDP 通常由以下 5 个元素组成：

（1）状态空间（State Space）：一个集合，通常用符号 S 表示，包含了问题中所有可能的状态。$s_t \in S$ 表示智能体在 t 时刻下的状态。

（2）动作空间（Action Space）：智能体可以执行的所有动作的集合，通常用 A 表示。

（3）状态转移概率（Transition Probability）函数：该函数描述了系统从一个状态转移到另一个状态的概率，通常用 P 表示。

（4）奖励函数（Reward Function）：用于评估每个状态转移的好坏程度，目标是最大化累积奖励，通常用 R 表示。每当系统从一个状态转移到另一个状态时，都会根据奖励函数获得一个奖励值。这个奖励值可以是正的（表示好的转移）或负的（表示不好的转移）。

（5）折扣因子（Discount Factor）：表示未来奖励的折扣程度，通常用γ表示其中 $0 \le \gamma \le 1$。较小的折扣因子意味着智能体更重视眼前的奖励，反之则意味着智能体更看中长远的利益。由此可见，γ决定了智能体如何权衡眼前奖励与未来的奖励。

马尔可夫决策过程的具体流程如图所示。在离散时刻 t，智能体与环境产生交互，智能体观察到此刻环境的状态 s_t ，通过策略π选择一个动作 $a_t = \pi(s_t)$ 执行。环境通过状态转移概率函数 P 变成一个新的状态 s_{t+1}，并根据奖励函数反馈给智能体一个奖励值 r_t。重复上述过程不断循环。当循环的次数是有限个的时候，则是一个有限 MDP。已知马尔可夫决策过程（S，A，P，R，γ），我们可以设定各种各样的策略π，而强化学习的目标就是从众多的策略中选择回报最大的策略。为了评价每种策略π的回报值，定义了累计回报 G，公式如下。

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{1}$$

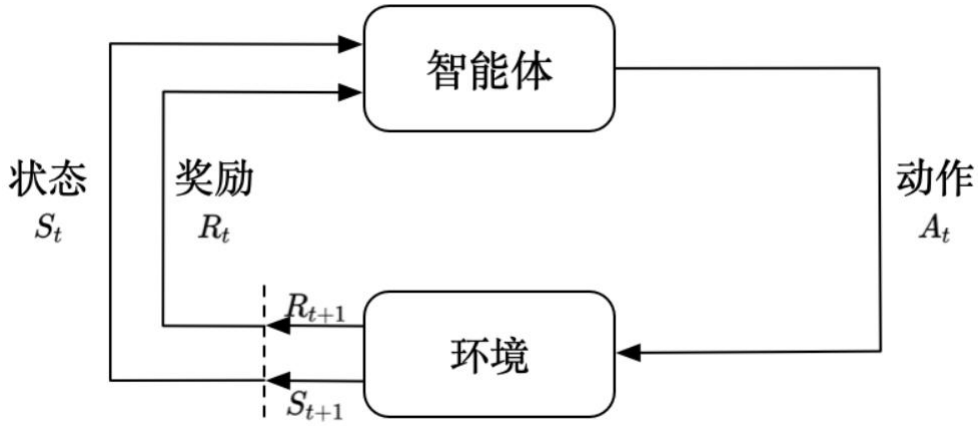

图 1 马尔可夫决策过程示意图

2.2 深度强化学习

2.2.1 基本概念

深度强化学习（Deep Reinforcement Learning，DRL）是深度学习与强化学习的结合。深度学习（Deep Learning，DL）通过模拟人脑神经网络的工作方式，构建深度神经网络模型，实现对大量数据的分析和处理。

相较于传统强化学习，深度强化学习的显著优势在于能够利用深度神经网络来处理高维输入，智能体因此有处理更为复杂的环境和更大数据量的能力。这一结合不仅扩展了强化学习的应用范围，还显著提升了其在处理复杂任务时的能力与效果。

2.2.2 深度 Q 网络

深度 Q 网络算法是深度强化学习的经典算法，是由 Mnih 等人提出的一种用于离散动作空间的无模型强化学习算法[3]。深度 Q 网络算法相对于 Q 学习算法的改进主要体现在:采用神经网络的方式来替换 Q 表，并使用梯度下降的方式更新神经网络的参数，从而对行为值函数（Q 值）进行值函数近似，这样解决了 Q 学习算法无法处理复杂的连续状态问题的能力。

然而，这样的设置带来了两个问题：1 是由于神经网络参数是从自身学习的，这样会导致训练的不稳定；2 是数据之间有较为强烈的关联性，在一定的程度上会影响网络训练的收敛速度与效果。

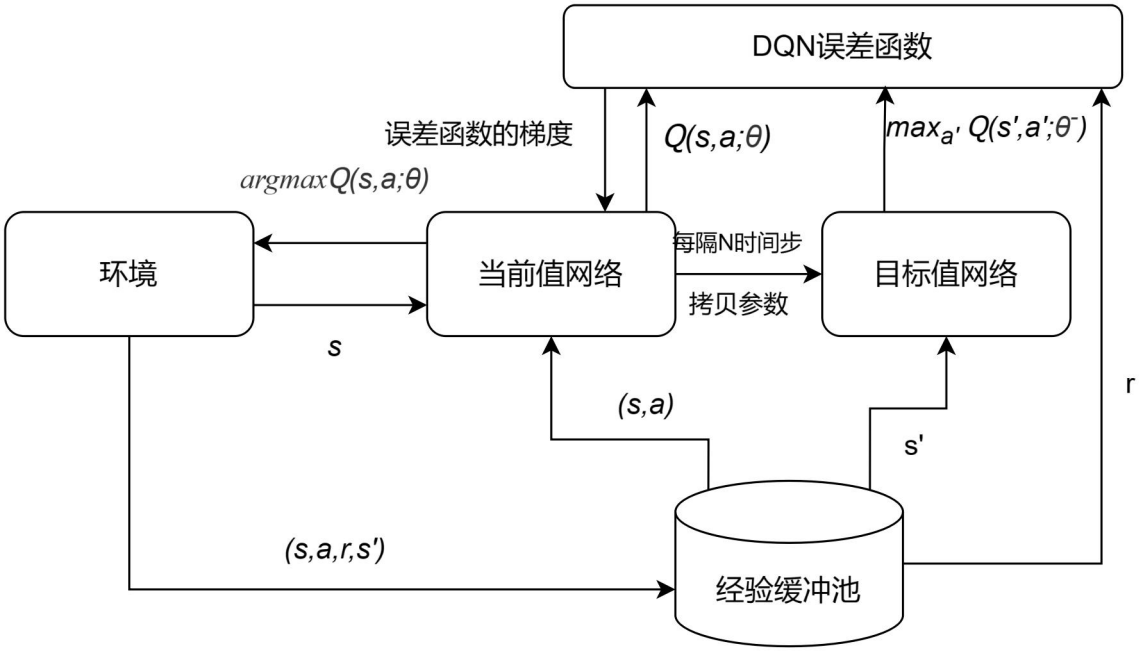

图 2 深度 Q 网络运行结构

为了解决上诉问题，在 Nature DQN 中引入了另外一个神经网络，目标值网络。不同于当前值网络每时每刻都在更新自己的参数，目标值网络只有在固定的时间间隔后才会复制主网络的参数来更新自己的参数从而实现学习的稳定。除此之外，还通过经验回放（Experience Replay）的方法：智能体与环境互动产生的数据放入经验缓冲池当中，神经网络的训练数据是从经验缓冲池中随机选取，这样一定程度上打乱了数据之间的关联性。最后，还为智能体制定了一个ε贪心策略，智能体有ε的概率随机选取动作，有 1 - ε的概率选择 Q 值最大的动作。在智能体探索的时期，ε往往是一个较大的数，这有助于智能体探索未知的环境，带来更多的经验；而在训练的后期则往往会减小ε的数值以稳定训练。DQN 的运行结构图就如图 2 所示。

2.3 数学公式表示

数学公式的表示在处理数学语言相关的任务中（例如数学公式检索）是十分重要的。在过去的几年里，诸如 GPT[14]、BERT [15]等大语言模型在自然语言处理（Natural Language Processing，NLP）的各种任务上取得了巨大的进步。受到这些成功案例的启发，在处理数学语言的过程中可以借鉴 NLP 领域的方法。

通过查阅资料发现，有人将数学公式认为是简单的文本，按照从左到右的符号的序列[16]。但是，不同于普通的文本，数学公式有着强烈的结构特性。为了体

现数学公式的结构特征，于是有人通过数据结构中的树（Tree）来表示数学公式，例如（1）Symbol Layout Tree （SLT），表示数学公式中的每一个数学符号所在的空间位置; （2）Operator Tree （OPT），定义了数学公式中各个操作符的计算顺序。

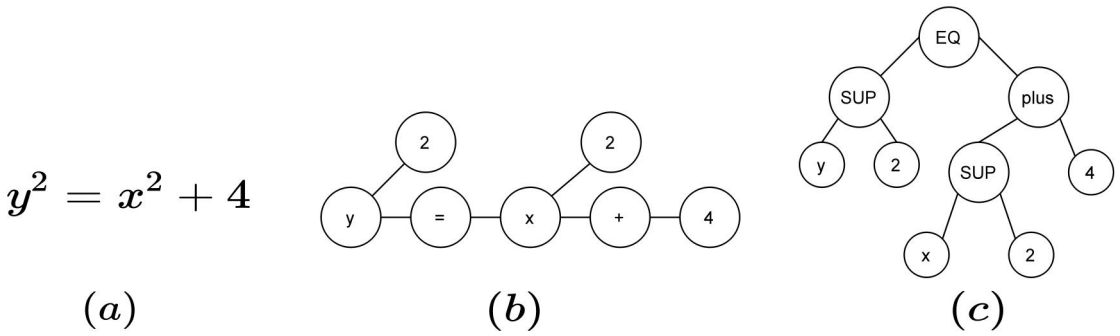

图 3 公式“$y^2 = x^2 + 4$”（a）和其对应的 SLT（b）和 OPT（c）的表示形态

图 3 中展示了数学公式“$y^2 = x^2 + 4$”（a）和其对应的 SLT（b）和 OPT（c）的表示形态。在表示数学公式的内容上，OPT 相较于 SLT 有着两个显著的优势。第一点，SLT 主要关注数学公式中符号的布局和排列。它侧重于符号之间的空间关系和显示顺序，而不是它们之间的数学运算关系。所以 SLT 更适用于排版和显示数学公式的场景。第二点，OPT 通过树形结构准确地反映了数学公式的内在逻辑和运算顺序[17]。每个节点代表一个运算符或操作数，节点之间的连接关系则体现了运算的层次和优先级。这种结构化的表示方式使得 OPT 能够精确地捕捉数学公式的所有细节和特征，便于进行表达式的解析和操作。除此之外，更重要的是 OPT 中包含着数学语法和语义，这些元素指导着数学公式运算的恢复过程[18]，也就是说 OPT 能够与原始公式之间进行无差别的互相转换，而且转换的过程中不会丢失任何信息或引入任何误差。

3. 方法设计与实现

3.1 深度强化学习框架

如前文 2.2.2 中介绍，DQN 是一个经典的深度强化学习算法，其在处理连续状态空间和离散动作空间的问题上有着显著优势，因此本文的算法框架将基于 DQN 实现。

DQN 算法中有 3 点独特的改进，分别是双网络结构、经验回放和ε贪心策略。本文在实现这三点上分别：

（1）双网络结构：初始化构建两个一模一的网络：当前值网络和目标值网络。当前值网络实时更新自身参数，而目标值网络按照指定时间步间隔从当前值网络复制参数来更新自身参数。

（2）经验回放：本文通过模拟实现一个经验池，当池子满了之后按照最新的覆盖最老的策略更新经验池，同时还实现了 Sample 接口，能够从经验池中随机采样，击破数据间的相关性。

（3）ε贪心策略：本文中ε并非是一个固定参数，而是根据智能体于环境互动的不同时期动态变化的。在初期，智能体注重探索，ε是接近 1 的；在末期，智能体注重稳定ε会很小但也不会是 0。

3.2 数学表达式表示方法的实现

如前文 2.3 中所述，OPT 在表示数学表达式时保留了数学表达式的结构和语义信息，因此本文采用 OPT 来表示数学表达式。本文主要参考借鉴了 MathGPT[19] 中针对数学语言部分的处理。在介绍如何将一个数学表达式转换成 OPT 的流程之前，先介绍 4 个在转换流程中会使用到的工具与项目，它们在转换过程起到了十分重要的作用。

（1）LaTeX：一种基于 TeX 的排版系统，由美国计算机学家莱斯利·兰伯特（Leslie Lamport）在 20 世纪 80 年代初期开发[20]，LaTeX 的设计目标之一是分离内容与格式，使得作者能够专注于内容创作，而无需过多关注版式设计，从而得到高质量排版的作品。由于其开源免费且在处理数学排版领域能力出众，许多数学公式都用 LaTeX 形式保存。

（2）数学标记语言（Mathematical Markup Language，MathML）：一种基

于 XML 的标准，用来在互联网上书写数学符号和公式的置标语言[21]。MathML 主要由两部分组成，分别是 Presentation MathML 和 Content MathML。前者是表现形式的 MathML，侧重于如何在布局上正确的显示一个数学表达式；后者是内容内涵的 MathML，侧重于表达式的语义与结构特征，而不单单是布局，这和 OPT 有着相同的目标，也正是本章关注的重点。

（3）LaTexml：一个用于将 LaTeX 文档转换为 XML 或 HTML 格式的工具[22]。它可以将 LaTeX 文档中的数学公式、图形和其他排版元素转换为 Web 友好的格式，以便在 Web 上进行展示或进一步处理。其在本文中主要用于将 LaTex 形式的数学表达式转换成一种类似于树结构的 Content MathML 形式。

（4）Tangent Combined FastText （Tangent-CFT）[23]：一个数学公式的嵌入模型，利用数学公式的两种层次化表示：SLT 和 OPT 来对数学公式进行嵌入，前者表示数学公式中符号的空间位置，后者则表示符号的语义信息。最后获取的公式嵌入同时捕获到了公式的空间结构信息和语义结构信息，在公式检索任务上取得了有效提升。在本文中主要用于将 Content MathML 形式的表达式转换成初始的 OPT 的形式来表示数学表达式。

本文中将输入的数学表达式转换成 OPT 的具体流程如图 4 所示。以函数的 LATEX 编码形式“x^{2} + 1.2x”为例，将 LATEX 代码通过 LaTexml 软件转换成对应的 Content MathML 形式（HTML 类型），再通过 Tangent-CFT 转换成初始的 OPT 形式（JSON 类型）。然后对初始的 OPT 进行特殊处理得到最终的 OPT 格式。特殊处理主要有两个部分：1.对于每一个 Operator 节点，在其子节点的末尾增加一个 E 节点表示结束，这有助于 Token 序列复原 OPT 格式。2.对于每一个 Number 节点，用 O^N 标识成特殊的 Operator 节点，然后将 Number 节点转换成一棵子树，每一个子节点是 Number 位数所对应的数字，包括小数点。例如图 2 中的“1.2”则转换成“1”、“.”、“2”三个子字符。通过这种方式，我们只需要“0”到“9”和“.”这 11 个符号就能表示所有的小数，避免了 Number 无穷个数的问题。最终，按照深度优先的顺序将 OPT 转换成一串 Token 序列用于表示数学表达式。

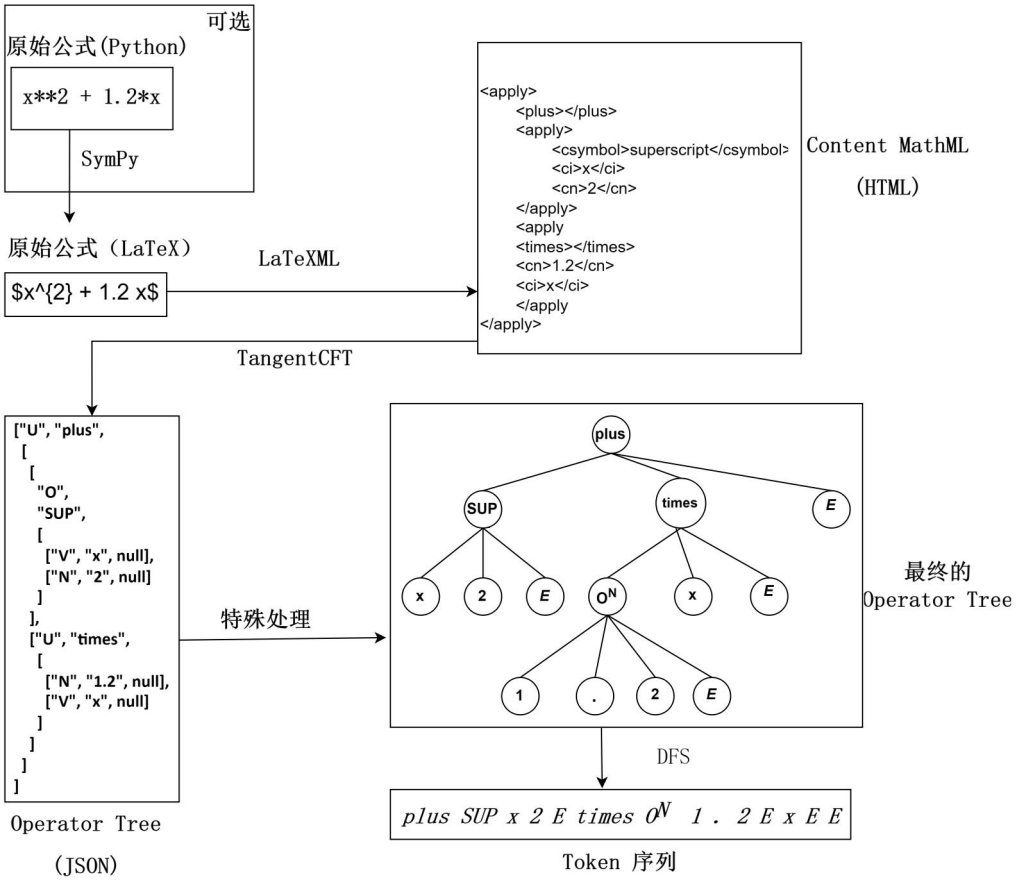

图 4 处理表达式“x2 + 1.2*x”的工作流程**

3.3 动作的实现

在深度强化学习中，动作扮演着至关重要的角色。在本文中，动作是智能体在特定状态下对函数进行的一系列操作，旨在降低函数的复杂程度、优化函数的表示和近似效果。我们定义了五种动作，分别是：括号展开、合并同类项、因式分解、换元和泰勒展开。

（1）括号展开：一种基本的数学操作，它不仅可以简化函数的表达形式，还可以体现函数内部的运算结构。

（2）合并同类项：合并相同的项，不仅可以降低函数的复杂度，还可以将要被处理的元素集中起来。

（3）因式分解：将复杂的函数表达式分解为简单的因子乘积形式，为换元操作提供了便利。

（4）换元：通过引入新的变量来简化函数或转换问题形式，常常在解决复杂问题时发挥关键作用。如何制定引入换元的策略对降低函数复杂度有着重要的

影响。本文的换元策略基于函数表达式是通过 OPT 表示的。假如树只有一层，说明此时的表达式可能只是一个个位数或者只是一个变量，没有还原的必要则不需要进行还原操作。反之，则遍历树的 root 的所有子树，将含有近似目标的子树加入候选列表。遍历候选列表中的元素，通过含变量的个数确定复杂度，变量数量越多则认为复杂度越高，最终选取复杂度最高的元素进行换元。例如图 5 中展示了换元操作下的表达式“sin(x)*cos(x) + sin(x) + 7”，root 是“+”操作符，每一个子树分别是“sin(x)*cos(x)”、“sin(x)”和“7”，所对应的复杂的分别是 2、1、0，则“sin(x)*cos(x)”将会被换元处理。

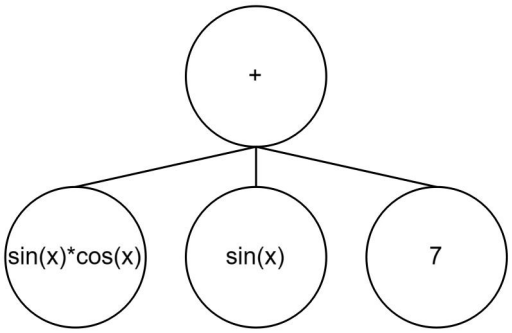

图 5 换元操作下的表达式“sin(x)*cos(x) + sin(x) + 7”

（5）泰勒展开：一种强大的近似工具，它可以将函数在某点附近展开成一系列幂次项的和，从而实现对函数的近似表示，这对消除 sin、log 等子函数，降低目标函数的复杂度有着重要作用。在本次实验中，泰勒展开往往和换元有着密切关系，前者会引入新的变量而后者往往会消除变量。

这五种动作相互补充、协同作用，共同构成了函数自动化近似过程中的操作集。在实际应用中，智能体将根据当前函数的状态和奖励函数的指导，选择合适的动作来优化函数的近似效果。通过不断地学习和优化，智能体能够逐渐掌握最佳的操作策略，实现更加精确的函数近似。

3.4 奖励函数的实现

奖励是对智能体在特定状态下采取特定动作后的及时反馈。奖励函数定义了每个状态下执行每个动作后所获得的奖励值，这些奖励值通常与智能体要达成的目标相关。智能体的目标通常是最大化长期累积的奖励，因此奖励函数在指导智能体的学习和优化过程中起着至关重要的作用。

本文中的奖励函数 F 接受输入状态 S1 和动作 A，并且最终返回状态 S1 情况下执行动作 A 后所获得的奖励值和一个用于判断新状态 S2 是否是任务结束状

态的布尔变量。

在深入探讨奖励函数的实现流程之前，首先介绍两个核心概念，这两个概念将有助于更好地理解接下来流程图中所呈现的内容。

（1）有效动作：状态 S1 通过动作 A 生成新的状态 S2，状态 S1 与 S2 的 Token 序列有差异，即状态 S1 和 S2 所对应的函数表达式不完全一样，则认为此时的动作 A 是一个有效动作。反之，则认为此时的动作 A 不是一个有效动作。例如，一个不含括号的函数表达式采用一个括号展开的动作。显然的，括号展开前后的函数将会完全一样，没有任何变化，则此时的括号展开就不是一个有效动作。

（2）非法动作：导致智能体陷入循环“刷分”的动作，这样的动作从单次动作的角度分析往往是一个有效动作，但是和前一个动作组合分析，就可能是一组无效动作。例如，状态 S1 通过动作 A1 生成新的状态 S2，然后状态 S2 又通过动作 A2 变成状态 S1，以此循环往复，不断的累计得分。即使这里的每一个动作都是有效动作但是对达成任务目标没有任何意义而且还会将智能体的训练引导向一个错误的方向。为避免这种“刷分”的情况，于是引入非法动作的概念。

本文中的奖励函数的具体流程图如图 6 所示。首先，判断输入的动作 A 是否是一个非法动作。如果动作 A 是一个非法动作则直接扣 1 分然后导致任务结束；如果动作 A 不是一个非法动作，再判断动作 A 是否是一个有效动作。如果动作 A 不是一个有效动作则直接扣 1 分然后导致任务结束；如果动作 A 是一个有效动作，加 0.1 分，然后再判断状态 S2 是否是任务结束状态的，即状态 S2 对应的函数表达式中的自变量是否有以三角函数和对数等形式的存在。如果状态 S2 不是任务结束状态的，直接加 0.1 分；如果状态 S2 是任务结束状态的，即状态 S1 通过动作 A 正常结束了任务，除了加有效动作的 0.1 分外，还将会得到一个额外的最终奖励。这个额外加分等于 $5*\sigma_1$（σ_1：原始函数 F1 与结束状态下的函数 F2 两者的近似程度）。为方便定量分析，本文在 0 到 1 之间（不包括 0 和 1）平均采样 20 个点，通过这 20 个点计算函数 F1 和 F2 的均方误差（Mean Squared Error，MSE），再将计算结果归一化，表示函数 F1 与函数 F2 两者的差异程度σ_2，最终通过 $1-\sigma_2$ 表示函数 F1 与 F2 两者的近似程度，即$\sigma_1 = 1 - \sigma_2$。

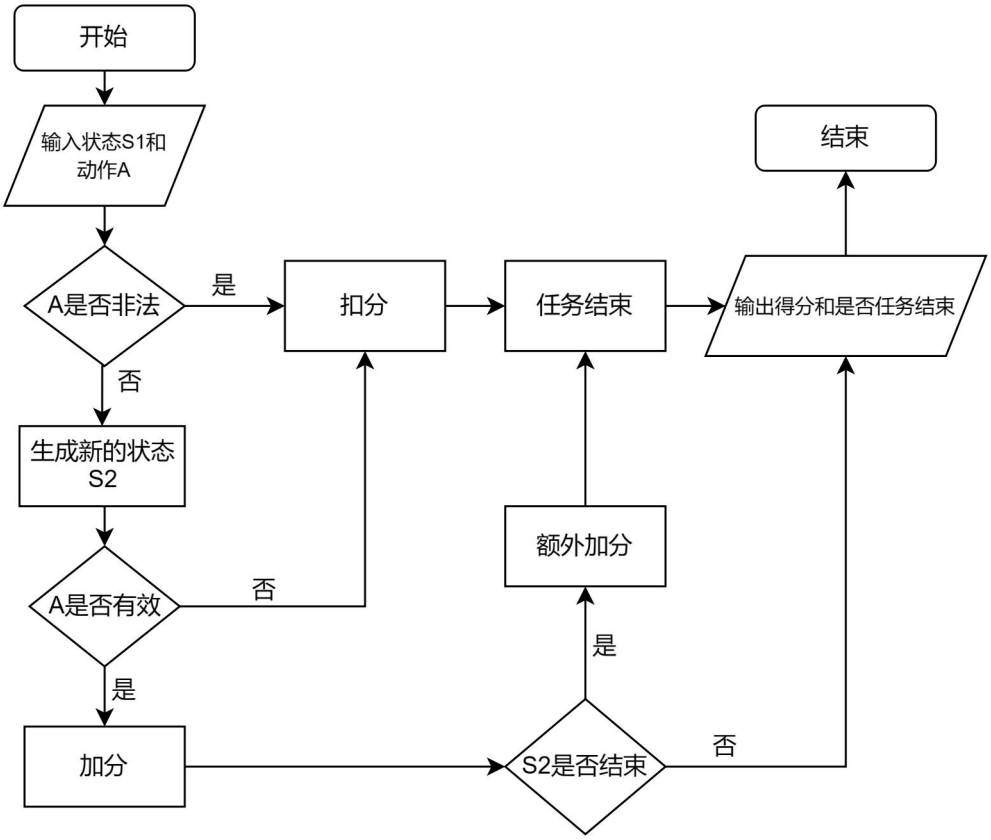
开始
输入状态S1和
动作A
A是否非法
是
扣分
任务结束
输出得分和是否任务结束
结束
否
生成新的状态
S2
A是否有效
否
额外加分
是
是
加分
S2是否结束
否

图 6 奖励部分的流程图

4. 实验结果分析

4.1 训练环境和超参数设置

本文基于 Python3.9 和 Pytorch 实现了第三章中介绍的所有内容。模型的所有训练都是在 Intel（R）Core（TM）i7-10875H CPU@2.30GHz 上进行的，未使用 GPU。深度 Q 网络算法中的超参数设置如表 1 所示：

表 1　DQN 超参数表

参数名称	参数值
学习率	2.5e-4
经验池容量	1000
每回合采样数量	128
折扣因子　γ	0.99 和 0.9
训练最大回合数	4000
训练开始回合数	400
训练频率	5
目标值网络更新频率	100
目标值网络更新率	1.0

DQN 中有两个网络分别是当前值网络（$Q_{current}$）和目标值网络（Q_{target}）。当前 Q 值是通过当前值网络计算；而目标 Q 值是通过目标值网络计算的，对应公式如下：

$$Q_{\text{target}}(s,a) = R + \gamma \cdot \max_{a'} Q(s',a') \tag{2}$$

其中 s’表示下一状态。DQN 训练目标就是想让当前 Q 值与目标 Q 值尽可能的相同，其中 DQN 的 Loss 函数是关于目标 Q 值与当前 Q 值间的均方误差，通过神经网络的自适应矩估计优化算法（Adaptive Moment Estimation，Adam）来更新网络参数。Loss 函数的公示如下：

$$L(\theta) = \frac{1}{N}\sum_{i}(Q_{\text{target}}(s_i,a_i) - Q_{\text{current}}(s_i,a_i))^2 \tag{3}$$

4.2 结果分析

本文的模型是在两个环境下同时进行训练的。在本文中，一个环境被认为是一个模版函数表达式通过调整表达式中的实数的大小和项的位置（不影响表达式的内容）产生的一系列表达式。公式 4 和 5 分别对应着环境一和二的模版表达式，其中 A、B 是随机生成的一位小数。

$$(A+\sin(x)^{*}\cos(x))^{*}(\log(x+1)+\sinh(x)+B) \tag{4}$$

$$A^{*}\cos(x)^{*}\sin(x)+\sin(x)^{*}(B^{*}\log(\mathrm{x})+\sin(\mathrm{x})) \tag{5}$$

在进行实验的过程中发现折扣因子γ会影响智能体的训练和最终的得分，本文的γ分为了两个版本：版本一的γ是 0.9 和版本二的γ是 0.99。

图 7 中展示了版本一和版本二的模型在训练过程中的 Loss 函数的下降变化：版本一模型的 Loss 函数收敛更加稳定，而版本二的 Loss 函数出现略微抖动。

图 8 中展示了版本一和版本二对应的模型在训练过程中预测的 Q 值曲线变化趋势，两者都收敛，其中版本一的 q 值收敛速度比版本二的快（前者在 2500 次左右，后者在 3300 次左右）。但是，版本二的收敛数值比版本一的更加高。

结合 Loss 曲线和 Q 值曲线分析，可见γ影响着模型训练：γ越接近 1，模型越难训练，但是最终累计收益往往更大。在任务目标并不是非常复杂的情景下，更高的γ虽然收敛速度慢了一点，但是相较于得分的提升来说是可以接受的，因此版本二是更加适用于本文的研究。

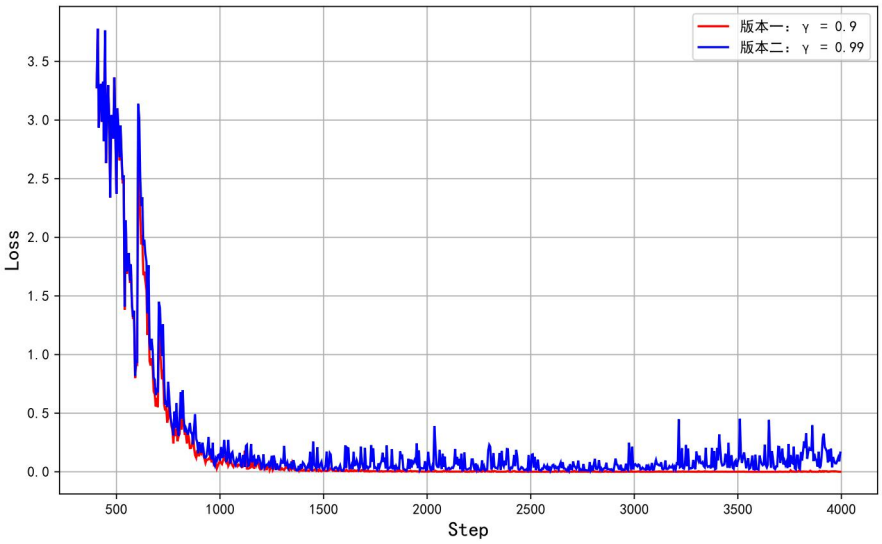

图 7 版本一和二的 Loss 曲线图

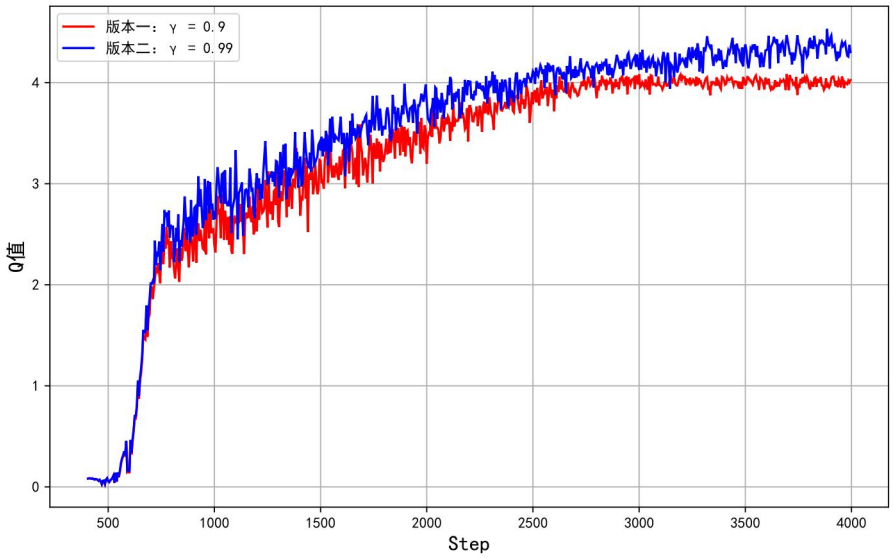

图 8 版本一和版本二的 Q 值变化曲线图

表 2 结果表

版本	场景一	场景二
一	4.06±0.13	4.11±0.03
二	5.04±0.30	4.91±0.15

表 2 中的数据是版本一和版本二在环境一和二下实际的累计得分的均值±标准偏差。两个版本都能够实现将原始表达式中关于自变量的三角函数和对数函数等近似成幂函数的目标，其中版本二的累计得分均高于版本一的。

图 9 中展示了γ是 0.99 的模型在实际处理表达式“(1.3*log(x)+sin(x))*sin(x) + 4*sin(x)*cos(x)”的简化显视过程，主要通过换元和泰勒展开策略实现数学表达式的函数的近似。

综上所述，训练过程和实验结果均成功验证了本文研究内容的可行性，其中γ是 0.99 的模型更适用于本文的研究。

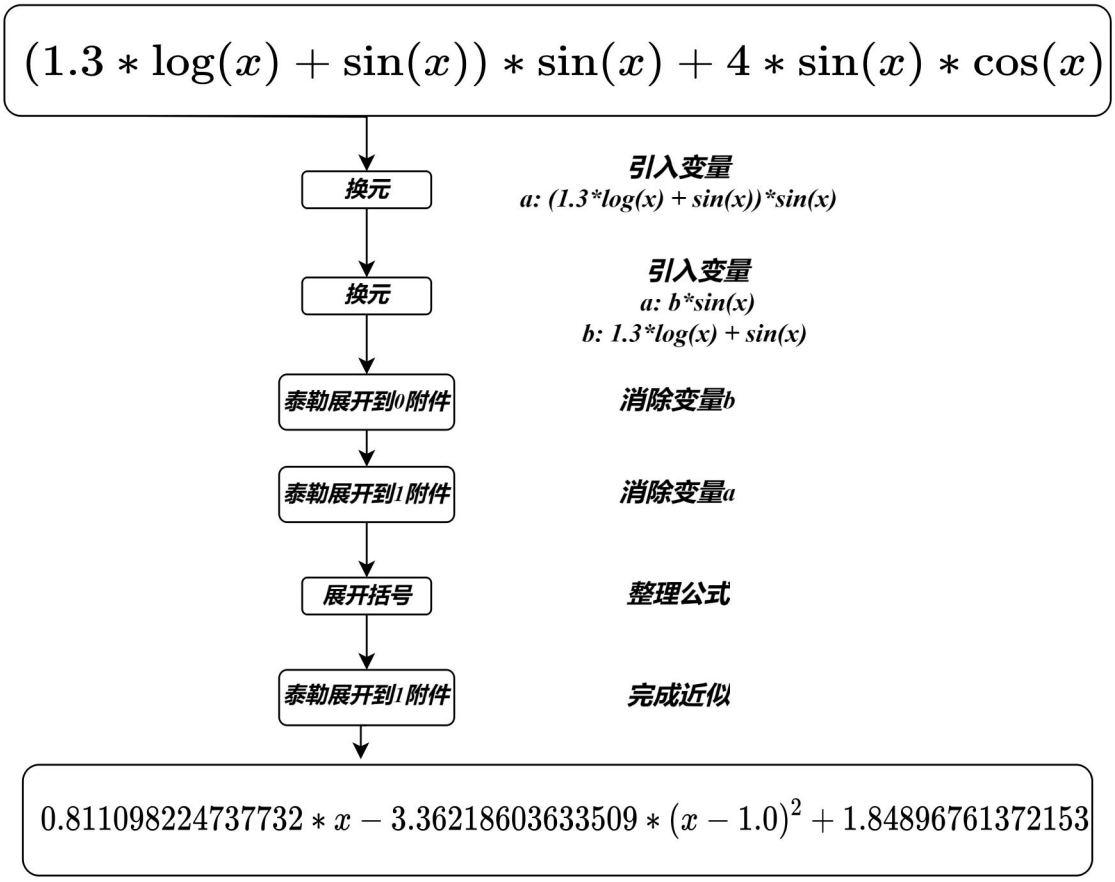

图 9 版本二模型近似流程显示图

5. 总结与展望

回顾全文，本文创新的将数学函数近似过程用马尔科夫决策过程建模，通过仿真实验验证了经典的深度强化学习算法 DQN 能够初步解决数学函数近似问题。在这个过程中，本文巧妙的借鉴 mathGPT 中数学语言处理的思路，将数学函数通过 OPT 表示。本文的研究为强化学习在数学语言处理领域提供了经验。

本文的研究存在一定的局限性，在诸多方面有着可以深入研究的地方。例如，DQN 算法虽然经典，但是已经有许多基于 DQN 改进的算法或其他深度强化学习算法，可以继续探究这些算法在本文的研究问题上的表现；加入更多的近似方法，在更多更复杂的实验环境中训练模型，探究通用性模型的可能。

参考文献

[1] 高阳,陈世福,陆鑫.强化学习研究综述[J].自动化学报,2004,(01):86-100.Doi:10.16383/j.Aas.2004.01.011.

[2] Bellman R. Dynamic Programming[J]. Science, 1966, 153(3731): 34-37.

[3] Watkins C J C H, Dayan P. Q-Learning[J]. Machine Learning, 1992, 8: 279-292.

[4] Sutton R S. Learning To Predict By The Methods of Temporal Differences[J]. Machine Learning, 1988, 3: 9-44.

[5] Mnih V, Kavukcuoglu K, Silver D, Et Al. Human-Level Control Through Deep Reinforcement Learning[J]. Nature, 2015, 518(7540): 529-533.

[6] 刘全,翟建伟,章宗长,等.深度强化学习综述[J].计算机学报,2018,41(01):1-27.

[7] 孙志军,薛磊,许阳明,等.深度学习研究综述[J].计算机应用研究,2012,29(08):2806-2810.

[8] Mcaleer S, Agostinelli F, Shmakov A, Et Al. Solving The Rubik's Cube Without Human Knowledge[J]. Arxiv Preprint Arxiv:1805.07470, 2018.

[9] Sun C, Kim D K, How J P. Romax: Certifiably Robust Deep Multiagent Reinforcement Learning Via Convex Relaxation[C]//2022 International Conference on Robotics and Automation (Icra). Ieee, 2022: 5503-5510.

[10] 王磊.基于深度强化学习的数学应用题自动求解器[D].电子科技大学,2019.

[11] 文森,钱力,胡懋地,等.基于大语言模型的问答技术研究进展综述[J/Ol].数据分析与知识现,1-17[2024-05-11].

[12] 温强.基于深度神经网络的机器人运动控制技术研究[D].燕山大学,2021.Doi:10.27440/d.Cnki.Gysdu.2021.000557.

[13] 董星辰.超密集蜂窝网络智能覆盖增强技术研究[D].电子科技大学,2020.Doi:10.27005/d.Cnki.Gdzku.2020.001827.

[14] Radford A, Narasimhan K, Salimans T, Et Al. Improving Language Understanding By Generative Pre-Training[J]. 2018.

[15] Devlin J, Chang M W, Lee K, Et Al. Bert: Pre-Training Of Deep Bidirectional Transformers For Language Understanding[J]. Arxiv Preprint Arxiv:1810.04805, 2018.

[16] Yuan K, Gao L, Wang Y, Et Al. A Mathematical Information Retrieval System Based On Rankboost[C]//Proceedings Of The 16th Acm/Ieee-Cs On Joint Conference On Digital

Libraries. 2016: 259-260.

[17] Peng S, Yuan K, Gao L, Et Al. Mathbert: A Pre-Trained Model For Mathematical Formula Understanding[J]. Arxiv Preprint Arxiv:2105.00377, 2021.

[18] Zanibbi R, Blostein D. Recognition And Retrieval Of Mathematical Expressions[J]. International Journal On Document Analysis And Recognition (Ijdar), 2012, 15: 331-357.

[19] Scarlatos A, Lan A. Tree-Based Representation And Generation Of Natural And Mathematical Language[J]. Arxiv Preprint Arxiv:2302.07974, 2023.

[20] 王勇，姚萍，王岚，等. Latex 与方正书版排版数学论文探讨[J]. 中国科技期刊研究，2012, 23(6): 1036.

[21] 唐亚伟.公式相似度算法及其在论文查重中的应用研究[D].渤海大学,2013.

[22] Ginev D, Miller B R. Latexml 2012-a Year Of Latexml[C]//Intelligent Computer Mathematics: Mkm, Calculemus, Dml, And Systems And Projects 2013, Held As Part Of Cicm 2013, Bath, Uk, July 8-12, 2013. Proceedings 6. Springer Berlin Heidelberg, 2013: 335-338.

[23] Mansouri B, Rohatgi S, Oard D W, Et Al. Tangent-Cft: An Embedding Model For Mathematical Formulas[C]//Proceedings Of The 2019 Acm Sigir International Conference On Theory Of Information Retrieval. 2019: 11-18.

致谢

我衷心感谢史玉回老师和赵琪老师在我进行本文的研究过程中提供的宝贵意见和建议。他们的专业知识和深入见解对我的研究方向和内容产生了深远的影响，让我能够更全面地考虑问题，并不断完善我的研究成果。在此，我向他们表达我最诚挚的感谢和敬意。