

摘 要

本文提出了一个新的东南大学 L^AT_EX 硕士研究生毕业论文模板，并说明了如何更优雅地写出一篇漂亮而无用的文章。

关键词： T_EX, L^AT_EX, 学位论文

Abstract

This article proposes a new Southeast University master degree thesis L^AT_EX template and explains how to elegantly write an article which is beautiful but full of shit.

Keywords: T_EX, L^AT_EX, Thesis

目 录

摘要	I
Abstract	III
第一章 绪论	1
1.1 研究工作的背景及意义	1
1.2 国内外研究	2
1.3 本文的贡献与创新	3
1.4 论文的结构安排	4
第二章 深度强化学习相关知识	5
2.1 强化学习	6
2.1.1 相关术语	6
2.1.2 基于价值的强化学习	6
2.1.3 基于策略的强化学习	7
2.1.4 价值与策略相结合的强化学习方法	8
2.2 深度学习	9
2.2.1 神经网络	9
2.2.2 激活函数	11
2.2.3 损失函数及其优化算法	13
2.3 深度强化学习	14
2.3.1 深度 Q 网络	14
2.3.2 近端策略优化	15
2.3.3 深度确定性策略梯度	16
2.3.4 深度强化学习算法的对比与选择	17
2.4 本章小结	18
第三章 仿真实验场景的设计与构建	19
3.1 城市交通仿真平台的可行性分析	19
3.1.1 VISSIM 介绍	19
3.1.2 SUMO 介绍	20
3.1.3 MATSim 介绍	21
3.1.4 仿真平台的选择	21

3.2	基于 SUMO 的城市交通仿真平台	23
3.2.1	平台设计目标	23
3.2.2	功能模块简介	23
3.3	实验场景的选择与搭建	24
3.3.1	路网的编辑与生成	24
3.3.2	出行模式的设计	26
3.3.3	流量的生成	29
3.4	本章小结	30
第四章	基于深度强化学习的出行模式与时间选择方法	31
4.1	马尔可夫决策过程框架	32
4.1.1	动作空间	32
4.1.2	状态空间	33
4.1.3	奖励函数	35
4.2	基于深度 Q 网络算法的模式与出发时间选择算法	36
4.2.1	神经网络结构设计	36
4.2.2	超参数的选择	38
4.2.3	模型的优化	40
4.3	基于聚类的深度强化学习方法	41
4.3.1	DBSCAN 聚类方法	42
4.3.2	聚类参数的选择	43
4.3.3	深度强化学习模型的改进	44
4.4	本章小结	45
第五章	模式与出发时间选择方法的训练与评估	47
5.1	模型的训练与分析	47
5.1.1	实验场景的准备	47
5.1.2	模型训练	48
5.1.3	训练结果分析	51
5.2	模型的评估	52
5.2.1	与传统方法的对比	53
5.2.2	模型参数的灵敏性分析	53
第六章	总结与展望	55
6.1	总结	55
6.2	展望	55
致 谢		57

参考文献 59

作者简介 61

第一章 绪论

1.1 研究工作的背景及意义

随着人口、就业和社会活动的增加，出行需求的增长往往是城市发展不可逆转的结果。这将导致一些大型城市地区的交通状况恶化，在高峰期时，一些主干道的通行能力将会低于出行的出行需求，出现拥堵现象。为了获得准确的出行需求预测并实施有效的需求管理策略，研究人员和政府机构了解出行者在出行时如何进行决策将会至关重要。一旦决策者知道出行者在何时何地以及将采取什么模式出行，就可以提供有效的解决方案来缓解拥堵。因此，出行决策建模成为交通研究的关键。

研究人员通常将出行决策描述为不同维度的备选方案选择，例如出发时间、目的地、方式和路线。这些选择问题通常被描述为离散或者连续选择模型。早期的出行决策模型只考虑了一个维度，即从该选择维度的一组相互排斥的备选方案中选择了一个备选方案。然而在实际生活中，需要结合不同行为维度的进行多维决策才足以支持日益增长的拥堵管理策略应用。

近年来，多维出行选择模型得到了更多的关注，因为与传统的一维选择模型不同，多维出行选择模型诠释了不同选择的相关性。在出行选择的问题中，模式选择和出行时间选择是两个非常重要的模块。从个人层面上看，这些都是出行者出行需求的重要性决策。在集计层面上，这决定了交通网络的荷载以及它的时空分布。不同交通方式的可行性和吸引性都取决于它的服务水平，如等待时间、出行时间、出行成本等，这可能会受到各种政策措施的影响，如高峰时段定价、拥堵定价、共乘或公交使用激励。对此类政策措施的评估需要一个出行模式和出发时间选择的综合框架。

模式选择与出行时间选择的研究大多基于随机效用最大化的离散选择模型。如嵌套 Logit 模型、交叉嵌套 Logit 模型和混合 Logit 模型可以应用于多维选择问题^[1]，因为它们具有建模不同选择维度之间相关性的能力。利用随机效用最大化的模型依靠着其强大的理论依据而被广泛地应用。基于随机效用的模型是可以解释出行选择的基本理论，而对于复杂的决策过程建模的适用性，尤其是在选择预测中，可能会受到随机效用函数中线性结构的限制。对于多维选择问题，不同维度之间的关联结构也需要预先确定。

效用的随机成分不仅可以解释出行者对与观察信息的局限性，而且可以考虑决策者的不完全信息和偏好的随机变化。然而，以下事实支持了对基于学习方法的出行选择模型的需求。首先，乘客在模式选择的决策过程，是由不同出行方式的服务水平信息告知和指导的。这些知识通常是通过各种方式获得的（包括出行经验），并且会随着时间动态变化。第二，出行决策受到一些行为因素的影响，其中乘客更倾向于（更少）选择（改变）他们已经习惯的模式。第三，交通系统的随机性和时间依赖性最可能引起出行者的自适

应模式切换决策，在这种决策中，出行者可能会根据以往的经验更新他们对每种出行模式的预期效用。传统方法不能解决决策过程中涉及的时间维度。因此，与传统的选择建模方法相比，基于学习的出行决策模型更可取。

近年来，强化学习因其强大的探索能力和自主学习能力，已经与监督学习、无监督学习并称为三大机器学习技术 [2]。伴随着深度学习的蓬勃发展，功能强大的深度强化学习算法层出不穷，已经广泛应用于游戏对抗、机器人控制、城市交通和商业活动等领域，并取得了令人瞩目的成绩 [3]。后续大量的研究成果也表明，强化学习是实现通用人工智能的关键步骤。

出行选择的决策过程是一个复杂的过程，会受到环境的影响而不断地变化，通过建立传统的出行选择模型来解释出行行为的方法过于理想化。而此类场景很好地契合了强化学习“无模型、自学习、数据驱动”，使用强化学习的方法可以将此类复杂的模型使用深度神经网络进行描述，通过提取不同外界环境的特征数据如等待时间、出行成本等构建状态输入，再对出行者的出行行为进行优化，利用大数据训练网络增加其真实性和可靠性。相较于传统的离散选择模型，强化学习的方法对复杂的场景适应能力有极大的提升，并且适用的场景更加广泛。

1.2 国内外研究

在出行选择的模型中，通常使用基于随机效用的离散选择模型对不同维度的选择行为进行建模。从 McFadden^[2] 在 1973 年提出了著名的 Multinomial Logit（MNL）模型用于行为选择建模以来，Logit 系列模型就被广泛应用于出行决策问题。然而，MNL 存在一个被公认的问题：它假设了不相关的替代方案的独立性，也被称为 IIA (independence of irrelevant alternatives) 特性。这表示替代方案未观察到的特征彼此之间相互独立，然而在一些出行选择的问题中这个假设将会不成立。例如，在离散出发时间选择中，相邻出发时间区间的未观测特征往往表现出显著的相关性。为了解决这一问题，Ben-Akiva^[5] 等在 1998 年提出了 Nested Logit（NL）模型和有序广义极值模型（Ordered Generalized Extreme Values, OGEV）。NL 模型能够识别嵌套组内不同替代方案之间的相关性。有序广义极值模型允许为每一对分组的备选方案提供一个相关参数。经过 Bhat^[6-8] 在 1998 年的测试，得出的结论是，NL 和 OGEV 模型的性能都优于 MNL。在此之后，不同的研究人员针对问题的多样性提出了更先进的 NL 模型，如 Ben-Akiva 和 Bierlaire^[9] 在 1999 年提出的 cross-nested Logit（CNL）模型和 Lemp^[10] 等在 2010 年提出的连续 CNL 模型。另一种改进的离散选择模型是 De Jong 等在 2003 年提出的 mixed Logit(MMNL) 模型^[3]，它通过改变 MNL 模型的参数随给定分布变化来考虑个体之间的异质性。然而，MMNL 的一个限制是，它需要对整个人口的参数分布进行特定的假设。这种限制可以通过潜在类 (LC) 模型来解决，该模型可以通过将总体划分为离散数量的类来捕获未观察到的偏好异质性^[4]。

另一种研究出行决策的主流方法是机器学习。与统计方法不同，在统计方法中，研

究人员试图确定模型结构和需要估计的参数，机器学习方法关注的是数据本身，并试图找到不同参数之间的关联。相较于随机效用的模型，机器学习模型的结构更加灵活，方便其探索不同特征之间的关联。针对出行决策的建模，主要有以下几种主流的机器学习方法：决策树模型 [11]，神经网络模型 [12]，以及支持向量机 [13]。与随机效用离散选择模型相比，这些机器学习方法可以处理大型数据库。然而，机器学习方法很少能捕捉到对出行行为研究较为重要的因素，包括时间价值 (VOT) 和弹性。此外，使用机器学习方法作为模型的主要框架还存在一个限制是机器学习模型对训练数据很敏感，在样本不足或有偏倚的情况下，会导致欠拟合或过拟合问题。

强化学习作为一种被广泛应用的学习机制，是利用环境的反馈评价作为学习的输入，学习主体拥有较强的环境适应能力的机器学习方法，因此适用于重复杂变的交通决策场景中。强化学习被用来解决各种领域的顺序决策问题，如机器人控制、电子游戏和系统优化等。强化学习的理论为人类行为提供了可解释的心理学和神经科学视角，即人类如何在给定的环境中计划自己的行为。此外，强化学习框架提供了智能决策的数学形式化形式，在智能体控制中具有强大而广泛的适用性，可直接应用于控制理论中顺序决策问题的求解。在交通领域，强化学习方法也受到了广泛的应用，例如交通流管理、自动驾驶，以及路线规划 [14]。近期，一些研究已经采用强化学习方法来建模出行者日常活动计划以及出行决策。

现有的出行决策与出行需求预测的研究工作多使用基于价值的强化学习方法，Janssens[15] 在 2007 年使用 Q-learning 的强化学习方法解决活动调度问题。Vanhulse[16] 等在 2009 年通过基于 Q-learning 的方法构建 MATSim 结构方程模型。Medhat[17] 等在 2008 年开发了一个更全面的动态公共交通路径和出行活动选择模型，称为 MILITRAS 系统，其中的模型使用了预先设定的奖励 (效用) 函数。

近几年，深度强化学习在控制复杂智能体的决策行为上取得了巨大的成功，并将强化学习算法与许多神经相关因素的研究相结合，激发了大量使用人工神经网络作为通用函数逼近器的强化学习方法的研究。Hausknecht[18] 等在 2016 年发表的著作研究了使用深度强化学习方法与多智能体合作行为。值得注意的是，它将多智能体研究中的矩阵博弈推广到更复杂的状态和行动空间。

1.3 本文的贡献与创新

在使用强化学习的仿真环境中，可以根据不同出行场景将出行者主要分成两种：有电子地图导航和无电子地图导航。在有电子地图导航的场景中，出行者信赖电子地图导航，会根据导航信息一般选择行程最短的模式和路径行驶。在此场景中，出行模式选择问题将转变为备选路径的行程时间预测和预估价问题。可以考虑使用预计到达时间 (ETA) 的计算方法解决 [22]。在无电子地图导航的场景中，出行者只能依靠自身过往经验，根据经验记忆选择效用最大的出行模式和路径。这种场景下，出行者的每次决策都会得到环境带来的不同反馈，与强化学习的思想相契合。因此这种场景可以使用强化学

习的模型解决。

相较于传统的离散选择模型，强化学习存在以下三点优势：

1. 强化学习模型直接与环境交互，减少了传统离散选择模型的假设限制。传统离散选择模型需要对环境的条件预先假设并检验，在复杂多变的环境下传统模型的弊端将会体现。

2. 强化学习的模型会减少采集数据的成本。一项新的交通政策在实施前需要大量的仿真验证，传统模型需要采集大量的现实数据来验证模型的有效性。强化学习可以基于智能体已知的场景，通过更改仿真环境中的基础设施或策略，使得智能体学习处理未知场景下的决策行为。

3. 考虑智能体记忆能力，贴近实际决策过程。在强化学习的模型中，智能体做出动作后会根据以往经验以及自身的探索不断优化不同决策行为的价值及策略，这与实际中人在进行决策时的惯性一致。

1.4 论文的结构安排

第二章 深度强化学习相关知识

在交通出行领域，如何合理地选择出行模式和时间，以达到高效、舒适、安全的出行，一直是研究者和决策者们关注的热点问题。传统的交通规划方法通常是基于流量预测和传统的数学模型来制定规划和决策，这种方法在一定程度上可以解决一些问题，但面临的挑战也日益增加。首先，传统方法很难处理复杂的交通场景和非线性的关系。其次，传统方法需要大量的数据和人工经验才能有效应对，但这些数据往往难以获取或者成本较高。此外，交通规划需要考虑的因素非常多，如出行模式、路线、时间等，而这些因素之间的复杂关系往往非常难以把握。

针对传统交通规划方法存在的问题，近年来，深度强化学习技术被引入到交通领域，成为了一种新的解决方案。深度强化学习通过学习交通系统的歷史数据，可以自动化地寻找规律和优化策略，以提高交通系统的效率和性能。通过引入深度强化学习技术，可以更好地解决交通出行领域中的一些问题。

对于模式选择问题，传统的方法主要是基于规则或者基于概率模型的方法。这些方法通常需要手动定义模型和规则，且模型和规则的适用性和可扩展性受到限制。而深度强化学习算法可以通过自主学习和适应环境的方式，学习到更加精准的出行模式选择策略，且不需要事先手动定义模型和规则。例如，可以通过深度强化学习来学习到乘客在不同时间、地点和情境下的出行偏好，以及在不同的出行模式之间做出选择的决策过程。

对于时间选择问题，传统的方法通常是基于历史数据或者基于概率模型的方法。这些方法存在着数据依赖性和模型精度的问题。而深度强化学习算法可以通过自主学习和适应环境的方式，学习到更加精准的出行时间选择策略，且不需要事先手动定义模型和规则。例如，可以通过深度强化学习来学习到乘客在不同时间、地点和情境下的出行偏好，以及在不同的出行时间之间做出选择的决策过程。

因此，引入深度强化学习算法可以有效地解决传统方法存在的问题，提高交通系统的智能化水平，优化交通出行效率，改善城市交通环境，为人们出行提供更加便捷、安全和可持续的选择。

本章主要将介绍了深度强化学习的相关知识。首先介绍强化学习的基本概念和相关术语，接着介绍深度强化学习基础知识中的深度学习，包括神经网络、激活函数、损失函数及其优化算法。然后详细介绍现在主流的深度强化学习算法，包括深度 Q 网络、近端策略优化、深度确定性策略梯度，并对这些算法进行了对比与选择。

2.1 强化学习

强化学习是一种基于马尔可夫决策过程的算法。在强化学习中，智能体根据环境状态和规定的策略进行交互，并根据环境给出的奖励信号产生新的状态。这个过程会不断循环，直到智能体完成设定的目标^[5]。强化学习算法利用产生的奖励数据来优化其行为策略，以获得最大的回报。本节将首先介绍强化学习算法的相关术语，然后根据智能体动作的选取方式，将强化学习方法分为基于价值、基于策略、以及基于价值和策略的三类方法，并对它们进行综述。

2.1.1 相关术语

智能体指的是具有独立思考能力且能够与环境进行交互的实体。在交通场景中，智能体可以是行人、车辆、信号灯等。

状态表示智能体对周围环境的感知，它是智能体感知历史的一个快照。所有状态的集合构成状态空间。

动作是智能体在某个状态下采取的行动。智能体可以采取的所有动作构成动作空间。

策略是智能体在当前状态下选择采取哪个动作的控制准则。它通常使用概率密度函数来表示，在每个状态下智能体采取各个动作的概率。

奖励是环境对智能体采取某个动作后的反馈效果。奖励可以为正向反馈或负向反馈。

回报是智能体从当前时刻开始采取行动到结束时所能获得的累积奖励之和。

状态转移是智能体采取某个动作后从当前状态转移到下一个状态的过程。状态转移通常具有随机性，这种随机性源自于环境^[6]。

2.1.2 基于价值的强化学习

基于价值的强化学习使智能体通过行动与奖励联系起来，通过试验和错误进行学习。智能体的主要目标是通过学习在不同情况下采取的最佳行动，随着时间的推移使其累积奖励最大化。在基于价值的强化学习中，智能体学习预测在特定状态下采取特定行动的价值。一个行动的价值通常被定义为智能体在特定状态下采取该行动并遵循特定政策所能获得的预期累积奖励。

在强化学习中，对于任意时刻 t ，在策略 π 下对状态 s_t 执行动作 a_t 会产生一个对应的奖励 R_t 。由于在强化学习研究背景下的问题具有马尔可夫性质，因此系统的总回报 U_t 与当前时刻的奖励 R_t 和未来时刻的奖励 R_{t+n} 有关。因此，可以表示为以下等式：

$$U_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots + \gamma^n R_{t+n} \quad (2.1)$$

式中， γ 是折减因子。

在 t 时刻的回报 U_t 中，未来的奖励是与未来的状态和动作相关，而两者都具有随机性，所以需要通过对 U_t 求解期望值 $Q(s_t, a_t)$ 来消除随机性^[7]。

$$Q(s_t, a_t) = E[U_t | S_t = s_t, A_t = a_t] \quad (2.2)$$

因此， $Q_\pi(s_t, a_t)$ 可以用来表示状态动作对 (s_t, a_t) 的价值。其中， $Q(s, a)$ 是强化学习中的动作价值函数。通过寻找在 t 时刻所有策略 π 中动作价值函数 Q_π 的最大值，可以获得最优策略 π 的动作价值函数 $Q^*(s_t, a_t)$ 。

$$Q^*(s_t, a_t) = \max Q(s_t, a_t) \quad (2.3)$$

对最优策略 π 中的动作集 A 取最大值，即可获取每一次的最优动作 a^* 。

$$a^* = \operatorname{argmax} Q^*(s_t, a_t) \quad (2.4)$$

在基于价值的强化学习模型中，其主要目的就是逼近最优的策略 π 的动作价值函数 $Q^*(s_t, a_t)$ 。可以利用神经网络等方法近似动作价值函数进行求解。

由式2.4中动作价值函数 $Q^*(s_t, a_t)$ 可以得到价值最高的动作空间 A^* 。在强化学习中，一般使用神经网络的方法近似函数 $Q^*(s_t, a_t)$ ，网络的输入为状态 s ，网络的输出为不同动作的价值。则有：

$$Q(s, a; \mathbf{w}) \rightarrow Q(s, a) \quad (2.5)$$

式中， \mathbf{w} 是价值网络 (Value Network) 的参数。可以通过不同状态下的奖励 R 利用时序差分算法更新价值网络，使得网络的参数 \mathbf{w} 更加精确。

$$Q(s, a; \mathbf{w}) \approx R_t + \gamma \cdot Q(s, a; \mathbf{w}) \quad (2.6)$$

最常见的基于价值的强化学习算法是 Q-learning。Q-learning 是一种估计最佳动作价值函数的无模型方法，它代表了智能体在特定状态下采取特定动作并遵循最佳策略所能获得的预期累积奖励。一个状态-行动对的 Q 值使用贝尔曼方程进行更新，该方程指出，一个状态-行动对的最佳 Q 值等于即时奖励加上折现的最大预期未来奖励。Q-learning 是一个迭代过程，包括在智能体采取每个行动后更新 Q 值，并接受奖励形式的反馈。随着时间的推移，智能体学会了所有状态-行动对的最佳 Q 值，使其能够在每个状态下选择最佳行动，使其累积奖励最大化。

基于价值的强化学习方法已经在各种应用中取得了巨大的成功，包括游戏、机器人和自动驾驶汽车，使得智能体能够学习如何在复杂和不确定的环境中做出最佳决策。

2.1.3 基于策略的强化学习

基于策略的强化学习主要是为智能体在环境中采取行动寻找最佳策略，以使奖励最大化。策略是一种从状态到行动的映射，它告诉智能体在特定状态下应采取何种行动。

基于策略的强化学习的目标是找到一个策略，使智能体的预期奖励在一段时间内最大化。在强化学习中，使用概率密度函数 $\pi(a | s)$ 来控制智能体在不同状态下的动作选取，即策略函数。策略函数的输入为当前 t 时刻的状态 s_t ，输出为所有动作的概率值。依据策略函数得到的概率值对所有动作随机抽样后，确定在状态 s_t 下进行的动作 a_t 。当使用神经网络的方法近似策略函数时，则有：

$$\begin{cases} \pi(a | s; \theta) \rightarrow \pi(a | s) \\ \sum_{a \in A} \pi(a | s; \theta) = 1 \end{cases} \quad (2.7)$$

式中， θ 是策略网络的参数。

通过式2.2，对 $Q_\pi(s_t, a_t)$ 求取期望，通过积分消除概率密度函数 $\pi(\bullet | s)$ 中的动作 A 可以得到状态价值函数 V_π ：

$$V_\pi(s_t) = E_A[Q_\pi(s_t, A)] \quad (2.8)$$

状态价值函数 $V_\pi(s_t)$ 只与当前策略 π 和状态 s_t 有关。因此，状态价值函数可以用来评价当前状态下不同策略的价值。如果是离散的动作空间，状态价值函数 $V_\pi(s_t)$ 可以写作：

$$V_\pi(s_t) = \sum_a \pi(a | s_t) \cdot Q_\pi(s_t, a) \quad (2.9)$$

如果是连续的动作空间，则使用积分形式代替连加求和。由于连续动作空间的研究较复杂，并且大多数可以离散化，因此之后均为离散动作空间下的状态价值函数。通过式2.7中策略网络近似得到的策略函数 $\pi(a | s_t; \theta)$ ，可以近似状态价值函数：

$$V(s; \theta) = \sum_a \pi(a | s; \theta) \cdot Q_\pi(s_t, a) \quad (2.10)$$

基于策略的方法通常使用随机梯度上升法来更新策略。策略 π 由一组参数表示，使用策略梯度定理等技术计算出相对于这些参数的预期奖励的梯度。然后使用梯度上升法更新参数，以改进策略。策略学习是通过学习式2.9中的参数 θ ，得到价值最高的策略。这个过程中需要通过不断地改进策略网络参数 θ 的使 $V(s; \theta)$ 的值达到最大值。因此，可以将式2.9中的 $V(s; \theta)$ 对状态空间 S 求期望，将目标函数转化为 $J(\theta)$ ：

$$J(\theta) = E_S[V(S; \theta)] \quad (2.11)$$

基于策略的强化学习的缺点之一是它的计算成本很高，因为策略通常由大量的参数表示。此外，策略有时会卡在局部最优处，这可能使它难以找到全局最优策略。总的来说，基于策略的强化学习是一种在复杂和动态环境中寻找最优策略的强大方法。

2.1.4 价值与策略相结合的强化学习方法

在强化学习中，将策略网络与价值网络同时训练更新的方法称为策略价值结合学习方法。其目的为使智能体通过策略网络做出的动作价值越来越高的同时，使得价值网络

对动作价值的评价越来越精准。在策略价值结合学习方法中，可以把策略网络当作行动者（actor），价值网络当作裁判（critic）。价值网络会对智能体通过策略网络做出的动作进行评价，帮助更新策略网络参数，

通过联立式2.5与式2.10，可以得到通过神经网络方法近似后的价值函数 $Q(s, a; \mathbf{w})$ 与策略函数 $\pi(a | s; \boldsymbol{\theta})$ 。因此，状态价值函数可以写作：

$$\begin{cases} V(s; \boldsymbol{\theta}, \mathbf{w}) = \sum_a \pi(a | s; \boldsymbol{\theta}) \cdot q(s, a; \mathbf{w}) \\ \sum_{a \in A} \pi(a | s; \boldsymbol{\theta}) = 1 \end{cases} \quad (2.12)$$

此时，可以把策略网络当作行动者（actor），价值网络当作批评者（critic）。价值网络会对智能体通过策略网络做出的动作进行评价，帮助更新策略网络参数，使其目标函数 $J(\boldsymbol{\theta})$ 的值更大。行动者和批评者根据奖励和估计的状态-行动值进行更新。批评者通过最小化估计值和真实值（奖励和下一个状态的估计值之和）之间的平均平方误差来更新其对状态行动值的估计。行动者以估计的状态行动值为指导，通过最大化预期收益（未来奖励的总和）来更新其政策。

与其他强化学习算法相比，通过学习策略和价值函数，价值与策略相结合的强化学习方法可以比基于策略的方法更快地收敛，比基于价值的方法更稳定。它还可以处理高维的状态和行动空间，并且可以在实时环境中在线学习。价值与策略相结合的强化学习方法结合了基于政策和基于价值的方法的优点，可以同时学习最优政策和最优价值函数。然而，该方法需要仔细调整学习率和其他超参数以确保稳定的学习，而且它可能存在收敛问题和价值函数估计的偏差。

2.2 深度学习

深度学习在强化学习中发挥了重要作用，其中神经网络作为深度学习的核心，被广泛应用于强化学习中的状态表示、策略和价值函数估计等任务中。激活函数和损失函数也在强化学习中扮演着重要角色，对于神经网络的训练和优化具有至关重要的作用。本节将介绍深度强化学习中深度学习的应用，重点讨论神经网络在强化学习中的应用、常见的激活函数和损失函数，以及针对深度强化学习的优化方法。

2.2.1 神经网络

神经网络是一种机器学习模型，其灵感来自于人脑的结构和功能。它是由多个相互连接的节点或神经元组织成层，并形成一个系统。每个神经元接收来自其他神经元的输入，处理输入数据后产生一个输出信号。然后一个层的输出被用作下一层的输入，直至最终层输出结果。神经网络在训练期间从数据中学习经验并调整神经元之间连接的权重。权重决定了神经元之间的连接强度，它们使用优化算法进行更新，以达到预测输出和实际输出之间的误差最小化。神经网络已被应用于广泛的场景中，包括图像和语音识

别、自然语言处理以及时间序列预测等。目前在深度学习领域内使用的主流神经网络结构有前馈神经网络、递归神经网络和卷积神经网络。

神经网络最常用的架构是前馈神经网络，其输入数据是沿同一个方向流经各层。前馈神经网络主体是由一个输入层、多个隐藏层和一个输出层组成。各个层的职责各不相同：输入层接收输入数据，输出层产生神经网络的最终输出，隐藏层负责学习输入数据的特征。输入层的每个神经元代表输入数据的一个特征，而输出层的每个神经元代表神经网络预测的一个类别或一个值。隐蔽层中每个神经元的输出是通过对输入和权重的线性组合来计算的，并加入一个偏置项。然后，输出通过一个激活函数，将非线性引入网络。

隐藏层中每个神经元的输出可以按以下方式计算：

$$\begin{cases} z = \mathbf{w} * \mathbf{x} + b \\ a = f(z) \end{cases} \quad (2.13)$$

其中 z 是输入和偏置的加权和， \mathbf{w} 是权重向量， \mathbf{x} 是输入向量， b 是偏置项， $f(x)$ 是激活函数， a 是神经元的输出。

隐藏层中每个神经元的输出被用作下一层的输入，在最后一层的输出是神经网络的最终输出。在训练过程中，神经网络通过调整神经元之间连接的权重和偏差，使预测输出和实际输出之间的差异最小。

反向传播是一种用于训练神经网络的算法，通过调整神经元之间连接的权重和偏置项来训练。它是一种基于梯度的优化算法，计算损失函数相对于权重和偏置的梯度，并按照负梯度的方向更新它们。其中，损失函数衡量的是预测输出和实际输出之间的误差。回归问题最常用的损失函数是平均平方误差，而分类问题则使用交叉熵损失。损失函数相对于权重和偏差的梯度可以用微积分的链式法则来计算。链式法则指出，一个复合函数的导数等于其组成部分的导数的乘积。在神经网络的背景下，链式法则被用来计算损失函数相对于每个神经元输出的导数，然后通过网络传播误差来调整权重和偏差。

损失函数对于神经元输出的梯度可以按以下方式计算。

$$\frac{\partial L}{\partial a} = \frac{\partial L}{\partial z} * \frac{\partial z}{\partial a} \quad (2.14)$$

其中 L 是损失函数， a 是神经元的输出， z 是输入和偏置项的加权和。

式2.14右边的第一项是损失函数相对于加权和的导数，可以用激活函数的导数来计算。第二项是加权和相对于神经元输出的导数，也就是权重向量。然后，损失函数相对于权重和偏置的梯度可以通过在神经网络中各神经层向后传播误差来计算。首先计算输出层的误差，然后使用链式法则通过隐藏层向后传播，最后使用计算出的梯度和学习率更新权重和偏置项。权重通常在训练前被随机初始化。学习率决定了权重和偏置更新的步长，通常使用试验和错误或网格搜索来选择。高的学习率可能会导致对最优权重的过度拟合，而低的学习率则会导致缓慢的收敛。

训练神经网络的挑战之一是过拟合，即模型学习到的数据过于适合训练数据而在新数据上表现不佳。当模型相对于可用于训练的数据量来说过于复杂时，就会出现过拟合。正则化技术可以用来防止过度拟合。最常用的正则化技术是 L1 和 L2 正则化。L1 正则化给损失函数增加了一个惩罚项，与权重的绝对值成正比，而 L2 正则化则是增加了一个惩罚项，与权重的平方成正比。惩罚项的作用是鼓励权重变小，这有助于防止过度拟合。

2.2.2 激活函数

激活函数是神经网络的一个关键组成部分，如果没有激活函数，神经网络本质上只是线性回归模型，这将严重限制其灵活性。激活函数是应用于神经网络中每个神经元的输出的函数，目的是在模型中引入非线性，这对于捕捉数据中的复杂模式来说是必要的。其计算过程主要是在神经元输入的加权和添加一个偏置项后，对结果进行非线性转换。之后，激活函数的输出值将被传递到网络的下一层作为输入数据。目前在深度学习中应用最广泛的激活函数有 Sigmoid 函数、ReLU 函数和 Softmax 函数。

Sigmoid 函数是一条平滑的 S 形曲线，接受任何输入并输出 0 到 1 之间的值。Sigmoid 函数的公式如下：

$$f(x) = \frac{1}{1 + e^{-x}} \quad (2.15)$$

其中 x 是该函数的输入。

Sigmoid 函数图像如图2-1所示。

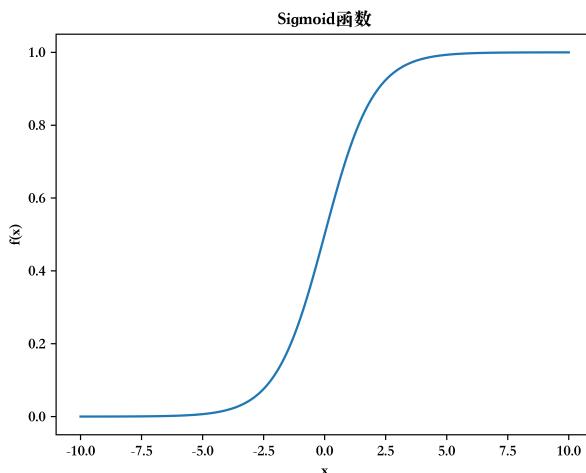


图 2-1 Sigmoid 函数图像

Sigmoid 函数是可微的，这意味着它的导数可以在任何一点处计算出来，这使得它很适合用于反向传播，也就是用于训练神经网络的算法。此外，Sigmoid 函数在 0 和 1 之间是有界的，这意味着它可以被解释为一个概率。然而，Sigmoid 函数也有一些缺点。Sigmoid 函数的主要问题之一是它存在梯度消失的问题。当 Sigmoid 函数的输入非常大

或非常小时，函数的输出分别变得非常接近于 0 或 1，函数的导数也会变得非常小，这可能导致梯度在反向传播期间消失。因为梯度在网络中向后传播时变得非常小，这样会使得网络难以学习更加深度的表征。

ReLU 函数是神经网络中另一个常用的激活函数。它是一个片状线性函数，接受任何输入，如果输入是正的，就输出，否则就是 0。ReLU 函数的公式如下：

$$f(x) = \max(0, x) \quad (2.16)$$

ReLU 函数图像如图2-2所示。

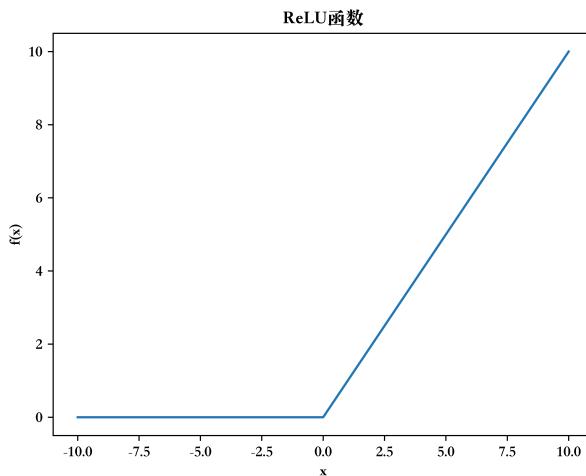


图 2-2 ReLU 函数图像

ReLU 函数的主要优点之一是它不存在梯度消失的问题。当 ReLU 函数的输入为正数时，该函数的导数为 1，这意味着在反向传播过程中，其计算出的梯度仍然很大。因为梯度在通过网络向后传播时不会消失，这使得网络更容易学习深度表征。ReLU 函数的另一个优点是它的计算效率高。由于该函数只是一个阈值操作，它可以用简单的逻辑运算来实现。然而，ReLU 函数的一个主要问题是，当 ReLU 函数的输入为负数时，该函数的输出为 0，这意味着神经元将会变得不活跃。这可能会导致整个神经元在训练过程中起不到任何作用，对网络的性能产生负面影响。

Softmax 函数是一种特殊的激活函数，常用于进行分类任务的神经网络的输出层。Softmax 函数在接受到一个输入矢量后，会输出一个和为 1 的数值矢量，可解释为概率。Softmax 函数图像如图2-3所示。

Softmax 函数由以下公式给出。

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \quad (2.17)$$

其中 x_i 是输入矢量的第 i 个元素，和是在矢量的所有元素上取的。

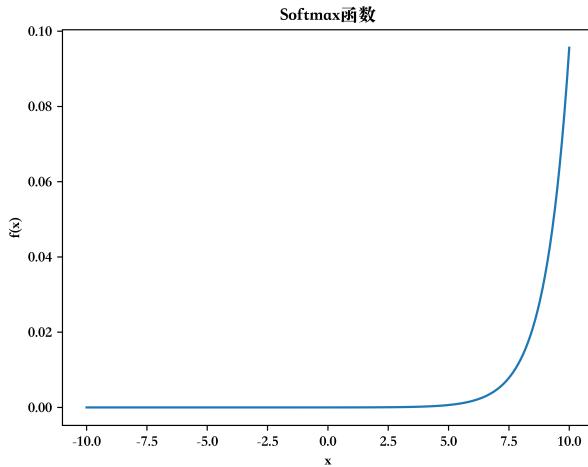


图 2-3 Softmax 函数图像

Softmax 函数的主要优势是可以确保网络的输出被解释为概率。这使得它非常适合用于分类任务，其目标是将输入分配到几个可能的类别中的一个。此外，Softmax 函数是可微分的，这意味着它可以用于反向传播来训练网络。

除了上面讨论的激活函数外，还有许多其他类型的激活函数被用于神经网络，包括双曲正切函数、指数线性单元（ELU）函数和缩放指数线性单元（SELU）函数等。这些激活函数都有自己的特性和使用场景，激活函数的选择取决于被解决的问题的具体需求，不同的激活函数可能更适合于不同类型的数据或任务。通过了解不同激活函数的特性，可以更好地设计神经网络，使其能够捕捉到实际数据中存在的复杂模式。

2.2.3 损失函数及其优化算法

损失函数是用来衡量神经网络的预测输出和实际输出之间差异的数学函数，其目标是提供一个衡量神经网络表现如何的标准。而损失函数优化是为了找到一组权重和偏置，使神经网络的预测输出与实际输出之间的差异最小。损失函数的选择取决于正在解决的具体问题。例如，对于回归问题，通常使用平均平方误差，而对于分类问题，通常使用交叉熵损失函数。

平均平方误差损失函数的公式如下：

$$L = \frac{1}{n} \cdot \sum_i y_i - \hat{y}_i^2 \quad (2.18)$$

其中 n 是数据集中的样本数， y_i 是第 i 个样本的实际输出， \hat{y}_i 是第 i 个样本的预测输出，和是在数据集中的所有样本中取的。

交叉熵损失函数的公式如下：

$$L = -\frac{1}{n} \cdot \sum_i y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i) \quad (2.19)$$

其中 y_i 是第 i 个样本的实际输出（二元分类为 0 或 1，多类分类为单次编码向量）， \hat{y}_i 是第 i 个样本的预测输出（0 和 1 之间的概率）， 和 是在数据集中的所有样本中取值。

神经网络最常用的优化算法是梯度下降法。梯度下降是一种迭代优化算法，它沿着损失函数的负梯度方向更新神经网络的权重和偏置。负梯度指向最陡峭的下降方向，这意味着沿着这个方向更新权重和偏置将导致损失函数值的减少。

梯度下降的更新规则如下：

$$w_i = w_i - \alpha \cdot \frac{dL}{dw_i} \quad (2.20)$$

其中 w_i 是第 i 个权重， α 是决定更新步长的超参数， dL/dw_i 是损失函数相对于第 i 个权重的偏导。

随机梯度下降（SGD）是梯度下降的一个变种，它基于随机选择的小型训练数据子集来更新权重和偏差。这样做的好处是在计算上比梯度下降法更有效率，因为它只需要计算一小部分数据的梯度。随机梯度下降的更新规则与梯度下降相似，但使用在数据子集上计算的梯度而不是整个数据集，更新规则：

$$w_i = w_i - \alpha \cdot \frac{dL_i}{dw_i} \quad (2.21)$$

其中 dL_i/dw_i 是损失函数相对于当前数据子集的第 i 个权重的偏导。

总之，损失函数优化是深度学习的一个关键方面，因为它直接影响到神经网络的性能。通过使用各种优化算法，如梯度下降、SGD 和 Adam，我们可以训练我们的神经网络来最小化损失函数，并在各种任务中获得高精确度。

2.3 深度强化学习

深度强化学习是强化学习中应用深度学习的一种重要方法，可以解决一系列复杂的任务。本节将介绍深度强化学习中常用的算法包括深度 Q 网络、近端策略优化和深度确定性策略梯度等。深度 Q 网络是基于 Q-learning 算法的一种深度学习算法，可以直接学习环境状态和行为之间的映射关系，以实现最优策略的学习。近端策略优化是一种基于梯度的方法，用于直接优化策略函数，以提高强化学习模型的性能。深度确定性策略梯度则是将近端策略优化与确定性策略相结合，以实现高效的连续动作控制。本节将深入探讨这些深度强化学习算法的原理和应用。

2.3.1 深度 Q 网络

深度 Q 网络（DQN）是基于 Q-learning 算法的深度强化学习算法，其目的是使用深度神经网络来近似给定状态-动作对的最佳动作价值函数。在 Q-learning 中，智能体通过选择最大化动作价值 Q 的行动来学习最大化其预期的未来回报， Q 值是在给定状态下采取行动并在之后遵循给定策略的预期未来回报。

在式2.2和式2.3中分别给出了动作价值和最佳动作价值的定义，深度Q网络算法使用一个深度神经网络来近似动作价值函数。该神经网络将当前状态作为输入，为每个可能的行动输出一个动作价值，智能体所选动作的动作价值被用来更新神经网络的权重。在深度Q网络算法中，下一个状态的动作价值是用目标网络来估计的，目标网络是一个具有固定权重的主网络的复制模型。目标Q值被用来更新主网络的权重，主网络被用来估计当前状态的动作价值。DQN中使用的平均平方误差损失函数，它用于衡量网络输出的Q值和目标Q值之间的差距。可根据式2.18得到用于更新神经网络权重的损失函数 $L(\theta)$ ：

$$L(\theta) = E \left[\left(r + \gamma \cdot \max_{a'} Q(s', a', \theta') - Q(s, a, \theta) \right)^2 \right] \quad (2.22)$$

其中， θ 是主网络的权重， θ' 是目标网络的权重， r 是在状态 s 下采取行动 a 后获得的即时奖励。

深度Q网络算法使用经验回放来提高学习效率和稳定性。经验回放存储了一个固定大小的经验缓冲区，神经网络通过从缓冲区中随机抽取经验进行训练。经验重放减少了连续经验之间的相关性，使学习过程更加有效。经验重放也有助于防止网络对最近的经验过度拟合。

深度Q网络算法使用 ε -greedy 探索来平衡对新行动的探索和对当前策略的利用。 ε -greedy 探索以 $1-\varepsilon$ 的概率选择具有最高Q值的行动，以 ε 的概率选择一个随机行动。

$$a_t = \begin{cases} \underset{a \in A}{\operatorname{argmax}} Q(s_t, a) & \text{当概率为 } 1 - \varepsilon, \\ \operatorname{rand}(a) & \text{当概率为 } \varepsilon. \end{cases} \quad (2.23)$$

总之，深度Q网络算法是一种被广泛使用的强化学习方法，用于训练强化学习问题中的动作价值函数。它通过使用神经网络来近似动作价值函数，解决了传统Q-learning算法的一些局限性，这使得它可以在类似的状态和行动中进行泛化。它还使用了一个经验重放缓冲器和一个目标网络来提高稳定性并防止过度拟合。DQN算法已被成功应用于各种具有挑战性的决策问题中，包括游戏、机器人的控制等。

2.3.2 近端策略优化

近端策略优化(PPO)是一种近年来备受关注的深度强化学习策略优化算法，属于策略梯度方法，这意味着它将从当前策略收集的经验中学习。PPO算法的基本原理是通过优化策略函数来寻找最优策略。通过式2.7可以得知，在强化学习中，策略函数 $\pi(a | s)$ 通常是一个映射函数，它将当前状态作为输入，输出对应的行动。近端策略优化算法通过反复迭代，不断更新策略函数，使其逐渐趋于最优。

该算法的核心思想是限制每次策略更新的大小，避免策略函数发生大幅度变化，导致训练不稳定。具体而言，PPO算法采用一种被称为“近端策略优化”的方法，通过在优化目标函数中增加一个约束项来限制每次策略更新的大小，这个约束项通常被称为“剪

切项”，它会限制新旧策略之间的差异，并确保每次策略更新的大小不超过一个预设的阈值，以确保优化的稳定性。在近端策略优化算法中，优化目标函数是最大化经过剪切后的期望优势函数，而不是最大化期望回报函数。这里的期望函数表示当前策略相对于旧策略的性能提升程度。在近端策略优化算法中，使用剪切方法来限制当前策略和旧策略之间的差异，其优化目标函数可以写作：

$$L_{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} A_t, \text{clip} \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1 - \epsilon, 1 + \epsilon \right) A_t \right) \right] \quad (2.24)$$

其中， θ 表示当前策略的参数， θ_{old} 表示旧策略的参数， $\pi_\theta(a_t|s_t)$ 表示在状态 s_t 下，当前策略选择动作 a_t 的概率， $\pi_{\theta_{\text{old}}}(a_t|s_t)$ 表示在状态 s_t 下，旧策略选择动作 a_t 的概率， A_t 表示在时刻 t 的优势函数，用于表示在给定状态 s_t 和行动 a_t 下，相比于平均水平的预期奖励，当前策略的表现优劣程度。具体地， A_t 的定义如下：

$$A_t = Q(s_t, a_t) - V(s_t) \quad (2.25)$$

式2.24中，目标函数 $L_{\text{CLIP}}(\theta)$ 的第一项表示策略更新的目标是最大化期望回报函数，第二项表示对策略更新进行剪切，确保新策略不会偏离原来的分布太远。通常来说， ϵ 的取值较小，可以取 0.1 或 0.2 等较小的数值。当 ϵ 取较小值时，第二项的影响较小，策略更新更倾向于最大化期望回报函数。

在近端策略优化算法中，更新价值函数的方法通常是通过均方误差（MSE）损失函数来实现，类似于式2.22。近端策略优化算法已经在多个实际应用场景中得到了广泛的应用，然而，近端策略优化算法也存在一些不足之处。例如，PPO 算法对于大规模离散动作空间的问题处理较为困难，同时其算法复杂度较高，需要消耗大量的计算资源。此外，近端策略优化算法在处理一些特殊场景下，如存在不确定性的环境、存在噪声的环境等，可能会出现训练不稳定的问题。

2.3.3 深度确定性策略梯度

深度确定性策略梯度（DDPG）算法是一种结合了深度神经网络和确定性策略梯度的强化学习算法，主要用于解决连续动作空间的问题，这类问题中，智能体需要在一个连续的动作空间中选择动作，因此传统的强化学习算法无法直接应用于该类问题。深度确定性策略梯度算法通过结合深度神经网络和确定性策略梯度的方法，解决了这一问题。与传统的 Q-Learning 算法相比，深度确定性策略梯度算法在处理连续动作空间问题时，可以直接输出动作值，而无需在离散动作空间中搜索最优动作。

深度确定性策略梯度算法的主要思路是通过 Actor-Critic 模型来学习动作值函数，同时通过确定性策略梯度的方法来更新策略函数。其主要由四个部分组成：策略网络、价值网络、经验回放缓存和目标网络。其中策略网络和价值网络都采用深度神经网络来进行参数化，在训练时首先需要通过经验回放缓存来收集一定数量的状态转移样本，然

后从中随机采样一批样本，用于网络的训练。策略网络的作用是输出在当前状态下最优的动作值。价值网络的作用是评估策略网络输出该动作值优劣的评估值，然后根据这个评估值计算出相应的策略梯度，最后通过反向传播算法更新策略网络的参数。经验回放缓存的作用是记录智能体在环境中的经验，并从中随机采样用于网络的训练。目标网络的作用是解决训练不稳定的问题，其参数是由价值网络参数每隔一段时间拷贝而来的。

深度确定性策略梯度算法的主要优点可以总结为以下几个方面。首先，深度确定性策略梯度算法采用 Actor-Critic 模型，可以直接输出动作值，因此可以直接处理连续动作空间问题。其次，深度确定性策略梯度算法引入目标网络和经验回放缓存技术，可以提高网络的训练稳定性。此外，深度确定性策略梯度算法使用深度神经网络处理高维状态空间问题，可以有效提取状态特征信息，提高智能体的决策效果。最后，深度确定性策略梯度算法结合了确定性策略梯度和 Q-Learning 的思想，可以同时处理具有连续动作空间和延迟奖励的问题。

然而，深度确定性策略梯度算法也存在一些缺点：1. 使用经验回放缓存和目标网络等技术来提高训练的稳定性的同时会导致了训练时间的增加。2. 深度确定性策略梯度算法有许多超参数需要调节，例如网络结构、学习率、优化器等，这些超参数的不同取值会影响算法的表现，参数调节较为复杂。3. 深度确定性策略梯度算法在处理连续动作空间问题时，通常需要引入噪声，但对噪声的选择和调节会对算法的表现产生较大影响，对噪声较为敏感。

2.3.4 深度强化学习算法的对比与选择

在解决出行模式和时间选择问题时，本章已经介绍了三种深度强化学习算法：深度 Q 网络、近端策略优化和深度确定性策略梯度。现在需要对这三种算法进行对比，并最终在出行模式与时间选择的场景下适用的方法。

深度 Q 网络算法是一种基于 Q-Learning 算法和深度神经网络的强化学习算法。其优点是训练速度快、稳定性高，而且适用于离散动作空间和连续动作空间。缺点是难以处理连续状态空间问题。

近端策略优化算法是一种基于策略梯度的强化学习算法。其优点是能够处理连续状态空间和连续动作空间问题，具有很好的表现。缺点是训练速度较慢，而且需要大量的超参数调节。在解决出行模式和时间选择问题中，我们的状态空间较大，因此 DQN 算法的表现可能会受到限制。

深度确定性策略梯度算法是一种基于 Actor-Critic 模型和深度神经网络的强化学习算法。其优点是能够直接处理连续动作空间问题和高维状态空间问题，而且能够处理具有延迟奖励的问题。缺点是训练时间较长，且对超参数的调节比较敏感。

表2.1总结并列出了三种算法的优缺点。

因为问题的场景是一个离散的动作空间，而且深度 Q 网络算法的训练速度快、稳定性高，适合处理这种场景。同时，考虑到算法的可解释性和易于实现性，深度 Q 网络

表 2.1 三种深度强化学习算法的对比

模型	优点	缺点
深度 Q 网络	训练速度快, 稳定性高, 可以处理高维状态空间和延迟奖励	只适用于离散动作空间, 对参数的选择比较敏感
近端策略优化	收敛速度快, 能够保持高样本效率	训练时间较长, 需要手动调整超参数
深度确定性策略梯度	处理高维状态空间和延迟奖励, 学习到高质量的策略	对参数的选择比较敏感, 难以处理高维状态空间

算法在这方面也具有优势。虽然近端策略优化算法在处理连续状态空间和连续动作空间问题上表现较好, 但在本文的问题中, 状态空间较大, 训练速度也较慢, 不太适合。深度确定性策略梯度算法可以处理连续动作空间和高维状态空间问题, 但运算时间成本较高, 且对超参数的调节比较敏感, 因此也不太适合处理本文的问题。因此, 综上所述, 针对出行模式和时间选择问题, 通过对三种主流的深度强化学习算法的优缺点分析, 最终选择深度 Q 网络算法作为本文解决问题的主要深度强化学习方法。

2.4 本章小结

本章主要介绍了深度强化学习相关的知识和技术。在2.1小节中首先详细介绍了强化学习中的相关术语, 包括智能体、状态、动作、策略、奖励、回报以及状态转移等, 这些概念是深度强化学习算法的基础。随后, 依据智能体选择策略的原理梳理了基于价值、基于策略以及基于价值和策略相结合的强化学习方法。这些方法各有特点和优缺点, 基于价值的方法主要学习状态值函数或状态动作值函数, 而基于策略的方法则直接学习策略函数, 基于价值和策略相结合的方法则既学习状态值函数又学习策略函数。在实际应用中, 需要根据问题的具体特点选择合适的方法来解决问题, 以获得最佳的效果。介绍深度强化学习之前, 在2.2小节简要介绍了深度学习的基本知识, 包括神经网络、激活函数以及损失函数和优化算法等, 这些知识对于理解深度强化学习算法是非常重要的。之后, 在2.3小节详细介绍了三种深度强化学习算法中常用的方法, 包括深度 Q 网络、近端策略优化、深度确定性策略梯度。这些方法使用深度神经网络来近似值函数或策略函数, 使得智能体可以在高维状态空间中进行学习和决策。最后, 通过对比了不同深度强化学习算法的优缺点, 选择了深度 Q 网络的方法作为研究出行模式和时间选择的主要方法。

第三章 仿真实验场景的设计与构建

仿真平台选择与仿真环境的搭建在交通研究中扮演着非常重要的角色。随着城市化进程的不断加速和交通需求的不断增长，城市交通系统的规划和管理变得越来越复杂和困难。传统的基于观察和统计数据的研究方法已经无法满足交通系统优化和决策的需求，因此交通仿真技术应运而生。

交通仿真通过计算机技术对交通系统进行建模、仿真和分析，以获得交通系统行为和性能方面的信息。通过仿真平台的选择和搭建，可以对不同的交通系统进行模拟和测试，以评估不同交通系统的性能和效果。仿真平台可以帮助交通研究者理解交通系统的复杂性和动态性，预测未来交通系统的发展趋势，并提出优化方案和决策建议。

在交通研究中，不同的仿真平台有着各自的优缺点和适用范围。例如，商业软件VISSIM 可以模拟更加复杂和精细的交通系统，但是需要购买授权和付费，对于一些小型和非营利性项目来说，成本较高。相比之下，SUMO 是一款开源的、免费的仿真平台，可以轻松地进行定制和扩展，因此被广泛应用于学术界和非营利组织中。

在仿真平台的搭建方面，需要根据具体的研究问题和场景进行选择和设计。例如，在仿真场景的选择方面，需要考虑到不同的交通模式、路段的特点和流量情况等因素。在路网的编辑和生成方面，需要考虑到交通规划和设计的要求，并对道路的连接、交叉口的设计等进行合理的规划和调整。在流量的生成方面，需要根据实际情况进行仿真和预测，以准确地反映交通系统的行为和性能。

第三章将主要介绍仿真实验场景的设计与构建，首先分析不同仿真平台的优缺点，以及在深度强化学习的出行模式和时间选择研究中为何选择 SUMO 作为仿真平台。接着，详细介绍基于 SUMO 仿真平台的城市交通仿真平台，包括平台设计目标、功能模块简介。最后，完成本文实验所需场景的选择与搭建、路网的编辑与生成、出行模式的设计和流量的生成等方面。

3.1 城市交通仿真平台的可行性分析

3.1.1 Vissim 介绍

VISSIM 是由德国 PTV (Planung Transport Verkehr AG) 公司开发的一款功能强大的交通模拟软件，广泛应用于交通规划、设计、管理等领域。其模拟能力可以涵盖各种复杂的交通场景，如城市道路、高速公路、交叉口、公共交通、交通流、交通信号控制、公共交通路线规划和车辆路径规划等。

VISSIM 的建模功能非常强大，用户可以使用图形用户界面轻松创建交通场景，包括道路、交叉口和公共交通路线等。用户还可以调整交通流率、车辆类型、行人流量和

其他参数，以建立一个真实的交通网络。智能交通生成器使用真实的交通数据，为一天中的不同时段、工作日和周末创建真实的交通流模式。同时，用户还可以根据自己的需要进行定制，调整交通量、速度和其他参数，以快速创建虚拟的交通场景。

VISSIM 的仿真模拟功能是其最核心的功能之一，用户可以通过 VISSIM 的仿真引擎高度准确地模拟各种交通场景，从简单的交叉口到复杂的城市网络。仿真引擎可以生成实时交通数据，如车辆速度、行驶时间和延迟时间等。用户可以对交通管理策略进行模拟，如交通信号控制、公共交通路线规划和车辆路径规划，并评估不同交通管理策略的性能，比较不同方案的效果，做出明智的决策。同时，用户可以通过虚拟实验，得出真实世界中的交通流动规律，从而更好地解决实际问题。

VISSIM 还提供了强大的分析功能，用户可以分析和可视化模拟结果，生成大量的图表和表格来分析交通流量、拥堵情况、车辆行驶时间、车辆速度和其他指标。分析功能还允许用户比较不同交通管理策略的性能，评估不同参数对交通流的影响。通过 VISSIM 的分析功能，用户可以更全面、更深入地了解交通流动规律，并进一步优化交通系统的设计和管理。

整体来说，VISSIM 是一款功能强大的交通模拟软件，可以广泛应用于交通规划、设计、管理等领域。VISSIM 的建模、仿真和分析功能都非常强大，可以帮助用户高度准确地模拟各种交通场景，评估交通管理策略的性能，并优化交通系统的设计和管理。作为交通专业人员、研究人员和政策制定者的重要工具，VISSIM 为实现更安全、更高效和可持续的未来交通系统做出了巨大的贡献。

3.1.2 SUMO 介绍

SUMO (Simulation of Urban Mobility) 是一款由德国科研机构 DLR 开发的开源交通仿真软件，广泛应用于城市交通规划、交通管理、交通研究等领域。该软件旨在提供高效、可扩展和高度自定义的交通仿真，以及良好的可视化和数据输出。SUMO 具有开放性、高可靠性、高可扩展性和良好的可视化效果等特点，成为交通仿真领域的重要工具之一。

SUMO 的基本功能包括建模、仿真技术和可视化。用户可以使用该软件创建虚拟交通网络，并定义网络中的道路布局、交通流和车辆类型。SUMO 使用微观交通模拟引擎，可以模拟各种交通场景，如车辆路线、公共交通和行人运动，并生成实时交通数据。用户可以通过 SUMO 的图形用户界面对模拟结果进行可视化，也可以生成各种图表和表格总结模拟结果。

除了基本功能外，SUMO 还提供定制功能、多模式仿真、优化功能、并行化和集成等高级功能。用户可以通过 SUMO 定义影响模拟的不同参数、自定义脚本、创建自定义的车辆模型，并将其导入到模拟中。该软件支持多模式模拟，如汽车、公交车、火车和自行车，用户可以评估不同交通方式的性能，并对不同的交通计划进行比较。SUMO 还提供优化功能，可以优化交通信号灯时间、车辆路线和公共交通时间表，找到最佳的交

通管理策略和交通计划。SUMO 提供并行化功能，允许用户在多个处理器上运行模拟，加快模拟时间，并模拟更大的交通网络。此外，SUMO 可以与其他软件工具集成，如交通流模型、地理信息系统（GIS）和数据分析工具，为用户提供一套全面的工具来模拟和分析各种交通情况。

总之，SUMO 是一个功能强大、用途广泛的模拟软件，为用户提供了一套全面的工具来模拟和分析各种交通情况。该软件支持多模式交通的模拟，包括汽车、公交车、自行车和行人，它还提供了一些高级功能，如交通需求生成、交通信号控制和车辆路由。SUMO 的灵活架构和开源代码库使其成为研究人员、交通专业人士和需要高度可定制模拟工具的决策者的热门选择。SUMO 有能力生成真实的交通数据，并提供对交通系统性能的洞察力，在设计、优化和评估交通系统方面发挥了关键作用，以实现更安全、更高效和可持续的未来。

3.1.3 MATSim 介绍

MATSim 是一款由瑞士苏黎世联邦理工学院（ETH Zurich）开发的开源交通仿真软件。该软件基于代理人建模，能够模拟不同交通方式的选择和相互作用，以及不同交通场景下的个人行为和整个交通系统的运行情况。此外，MATSim 提供了全面的可视化工具，使用户可以实时查看模拟的结果，包括交通流量、旅行时间等重要指标。

MATSim 的模块化架构使用户能够根据自己的需求定制软件，以模拟新的运输系统或包括新的功能。这使该软件能够适应不同交通场景的具体要求，并成为研究人员和交通专业人士的热门选择。MATSim 还具有开源代码库的特点，允许用户修改软件并将其分发给其他人，使研究人员和交通专业人士能够合作和分享他们的工作。

MATSim 是一个功能强大的交通仿真软件，能够模拟多种交通方式和不同的交通场景，使用户能够评估不同交通系统的性能，并优化城市或区域内不同交通方式的使用。该软件已被广泛测试和验证，用于对全球不同城市和地区的交通系统进行建模和模拟，证明了其对交通系统进行精确建模和模拟的能力。MATSim 在设计、优化和评估交通系统以实现更安全、更高效和可持续的未来方面发挥了关键作用。

总之，MATSim 是一款功能强大、用途广泛的交通仿真软件，为用户提供了一套全面的工具来模拟和仿真不同的交通场景。该软件的模块化架构、多模式模拟能力、先进的可视化功能和开源代码库使其成为研究人员和交通专业人士的热门选择。MATSim 能够准确地模拟交通系统并评估不同交通方案的性能，在设计、优化和评估交通系统以实现更安全、更高效和可持续的未来方面发挥了关键作用。

3.1.4 仿真平台的选择

通过对三种主流交通仿真平台的介绍，表3.1总结梳理了各个仿真平台的优缺点。

SUMO、VISSIM 和 MATSim 是三款常用的交通仿真软件，它们各自具有一定的优劣势。其中，SUMO 是一款开源软件，用户可以免费使用和修改，而 VISSIM 则是商业

表 3.1 不同仿真软件的对比

仿真软件	优点	缺点
VISSIM	城市和区域交通系统、交通行为建模、支持多种交通模式、交通流量和排放量测量评估	商业软件、非机动车和行人交通模式处理能力有限、复杂交通场景模拟效果可能不够准确、缺乏开源模型和工具
SUMO	速度快、处理大规模交通网络、开放性、开源模型和工具、多种交通模式和路段可调整性	交通行为建模较为简单、非机动车和行人交通模式处理能力有限、缺少全面的可视化工具
MATSim	模拟多种交通方式和不同的交通场景、模块化架构、提供全面的可视化工具、开源代码库	对于大规模交通网络的处理能力有限、个人行为的细节模拟较为复杂、某些交通模式和场景的支持仍不够全面、模拟速度相对较慢

软件，需要购买授权才能使用。**MATSim** 虽然是开源软件，但其复杂度和使用难度相对较高。因此，使用 **SUMO** 平台作为仿真工具，能够在节约成本的前提下实现高质量的仿真。

在仿真速度方面，**SUMO** 的速度相对较快，适合处理大规模交通网络。而 **VISSIM** 的仿真速度较慢，在处理大规模网络时效果不如 **SUMO**。**MATSim** 在处理大规模网络时也存在一定的限制。因此，在大规模交通仿真方面，使用 **SUMO** 平台可以更快速地生成模拟结果，为交通规划和管理提供更快捷的决策支持。

SUMO 还提供了可定制的 API 和开源模型，便于进行二次开发和扩展，而 **VISSIM** 和 **MATSim** 的 API 和模型相对较为封闭，用户的自定义程度较低。这一特点使得 **SUMO** 平台能够更好地适应不同仿真需求，提高仿真的灵活性和可扩展性。同时，这也使得 **SUMO** 平台成为开发新的交通仿真工具和应用的理想平台，为深度强化学习等新兴技术的应用提供了广阔的发展空间。

除此之外，**SUMO** 对非机动车和行人等非机动交通模式的处理能力相对较强，而 **VISSIM** 和 **MATSim** 在这方面存在一定的局限性。这意味着 **SUMO** 平台可以更好地模拟现实交通场景，更准确地预测交通行为和流量，从而为交通规划和管理提供更高效的支持。另外，**SUMO** 平台上已经有相关的研究和开源工具，支持深度强化学习算法的应用。而在 **VISSIM** 和 **MATSim** 上的应用研究相对较少，缺乏相应的成熟工具和开源库。因此，使用 **SUMO** 平台能够方便地借鉴和复用这些成果，提高仿真的效率和准确性。

综上所述，SUMO 平台作为一款开源、高效、灵活和可扩展的仿真工具，非常适合用于本文的基于深度强化学习的出行模式和时间选择仿真。

3.2 基于 SUMO 的城市交通仿真平台

在前一节中，介绍了仿真平台的选择，重点比较了 SUMO、VISSIM 和 MATSim 三个交通仿真软件的优缺点，并说明了为何在基于深度强化学习的出行模式和时间选择仿真中，使用 SUMO 作为仿真平台是一个较为理想的选择。本节将进一步介绍基于 SUMO 的城市交通仿真平台的设计与构建，其中包含了平台设计目标和功能模块简介两个方面。

3.2.1 平台设计目标

本研究的目的是利用深度强化学习方法研究出行者的出行模式和时间选择行为，并利用出行者的学习行为，实现提高出行效率。为此，我们采用 SUMO 仿真平台作为我们的仿真工具，以模拟不同的交通环境和交通管理策略。

SUMO 仿真平台是一个开源的、高度可定制的交通流量仿真器，支持多种车辆类型和行驶策略，如私家车、公共交通、自行车和行人等。它还提供了一个完整的仿真工具链，包括道路网络编辑器 Netedit、仿真器 SUMO、交互式仿真器 SUMO-GUI 和命令行接口 TraCI 等，支持多种路线选择算法和交通灯控制算法，用户可以根据他们的应用场景选择最适合的算法。然而，如果开发者直接在实验流中调用 TraCI 而不改进其上的结构以满足实际需求，将会导致代码结构混乱，代码可读性、复用性和扩展性较差等问题，这种方案只适用于简单试验性质的仿真，不能用于系统性的研究工作。因此，SUMO 仿真平台的设计目标是为强化学习算法训练和调试不同结构的区域路网提供支持，能够灵活地构造和修改仿真实验的网络道路拓扑结构和环境参数，复现经典深度强化学习算法并提供实验算法流程示例，同时对实时实验和模型数据具有一定的记录和可视化功能。

3.2.2 功能模块简介

SUMO 包括 Netedit、TraCI、SUMO-Tools 和 SUMO-Plugins 四个重要的功能模块。Netedit 允许用户创建、编辑和管理道路网络，TraCI 允许用户控制车辆和路口的运动和行为。SUMO-Tools 用于模拟和评估交通管理策略，SUMO-Plugins 用于扩展 SUMO 的功能和行为。这些功能模块和工具可以相互集成，为用户提供了一个强大和灵活的仿真环境，使得用户可以模拟不同的交通场景和交通管理策略，评估其效果和优化方案，进而提高交通流量的效率和可持续性。

Netedit 是 SUMO 中一个重要的功能模块，用于创建、编辑和管理道路网络。用户可以通过 Netedit 提供的图形界面添加和编辑道路、路口、车道和交通灯等元素，以模拟不同的交通环境和交通管理策略。同时，Netedit 支持多种地图格式，如 OpenStreetMap、

Google Maps 和 Bing Maps 等。用户可以通过拖放操作和线条绘制工具轻松绘制道路网络。Netedit 还提供一些有用的工具，如道路长度和宽度测量工具，帮助用户更准确地绘制道路网络。创建和编辑完成后，用户可以导出 Netedit 文件以供 SUMO-GUI 或其他仿真器使用。Netedit 还支持 Python 脚本，用户可以通过编写脚本自动化地创建和编辑道路网络。

TraCI 是 SUMO 中的另一个重要功能模块，它允许用户控制车辆和路口的运动和行为，以模拟不同的交通环境和交通管理策略。TraCI 提供了一系列 API 接口，用户可以通过 Python 脚本访问和修改仿真器中的车辆和路口状态，如车辆的位置、速度、目标路段和加速度等信息。用户也可以控制车辆的加速、刹车和转向，以及路口的红绿灯和车辆进出等操作。TraCI 还支持多种车辆类型和行驶策略，并提供有用的工具和功能，如路网查询、车辆生成和路由规划等，帮助用户更好地控制和管理交通流量。TraCI 可以与其他 SUMO 模块和工具集成，如 SUMO-GUI、SUMO-Tools 和 SUMO-Plugins 等，为用户提供更多的定制和扩展能力。

SUMO-Tools 是 SUMO 中用于模拟和评估交通管理策略的功能模块，如交通灯控制和路由优化。SUMO-Tools 包括一个流量分析器、一个行驶速度分析器和一个决策支持工具等，用户可以使用它们分析和比较不同的交通管理策略，以评估其效果和优化方案。SUMO-Plugins 是 SUMO 的可扩展模块，允许用户添加自定义的功能和行为，如新的车辆类型、路线选择器、行驶策略和交通管理策略等。SUMO-Plugins 可以是 Python 脚本、C++ 插件或 Java 插件等，支持多种路线选择算法和交通灯控制算法。SUMO-Tools 和 SUMO-Plugins 可以与其他 SUMO 模块和工具集成，如 SUMO-GUI 和 TraCI 等，为用户提供更多的定制和扩展能力。使用 SUMO-Tools 和 SUMO-Plugins，用户可以模拟和评估不同的交通管理策略，创建复杂的交通场景，并添加自定义的功能和行为，以满足特定应用程序的需要。SUMO-Tools 和 SUMO-Plugins 为用户提供了一个灵活的仿真环境，使得用户可以针对特定的交通问题和应用场景进行定制和扩展。

3.3 实验场景的选择与搭建

在基于 SUMO 的城市交通仿真中，实验场景的选择和搭建是十分重要的，它直接决定了仿真结果的可靠性和实用性。实验场景的选择和搭建包括了路网的编辑与生成、出行模式的设计和流量的生成等多个方面。在本节中，将详细介绍如何通过 SUMO 的功能模块和工具，搭建出逼近真实世界的仿真场景，从而进行有效的交通流量仿真和出行模式探究。

3.3.1 路网的编辑与生成

在本研究中，我们使用 SUMO 作为交通仿真引擎。SUMO 是一个开源的交通仿真软件，支持建模、仿真和分析各种交通场景，包括道路交通、公共交通、自行车交通和

行人交通等。

为了研究基于深度强化学习的出行模式和时间选择，我们需要建立一个仿真环境来模拟交通流。这个仿真环境需要选择一个具有典型城市交通结构和各种出行方式的区域进行研究。因此，我们选择了中国苏州市的一个城市区域作为我们的研究对象，该区域面积大约为 20 平方公里，具有复杂的道路结构和丰富的出行方式，是一个理想的研究对象。图3-1是仿真场景选取的现实路网区域。



图 3-1 仿真场景选取区域

接着通过 OpenStreetMap (OSM) 来获取苏州市的路网几何和配置信息。OSM 是一个开源的地图服务，可以提供全球范围内的地图数据，包括道路、建筑和自然环境等。我们使用 OSM 提供的数据，利用 SUMO 软件进行仿真。在建立仿真环境时，我们考虑到苏州市的交通网络的实际情况，并进行了一些修正和调整。

SUMO 中的 netedit 模块是一个可视化的工具，用于编辑和修改道路网络。使用其添加一些缺失的路段和交叉口，还可以设置道路的形状和方向，以确保它与现有道路网络的拓扑结构相匹配，图3-2展示了在 netedit 模块中补齐 OSM 数据路网中缺失道路的场景。在实际道路网络中，路口通常是复杂的，并且需要进行精细调整才能更好地反映真实交通环境。可以在 Netedit 模块中设置路口的属性信息，图3-3展示了对交叉口的精细化操作界面，包括路口类型、交通信号灯、路口转向规则和优先级等。此外，还添加了一些交通规则和限制条件，如车速限制、交通信号灯和路口优先级等，以更真实地模拟交通环境。

图3-4是对选取区域使用 Netedit 模块搭建后的 SUMO 路网，包含 2,423 个节点和 4,970 条路段。

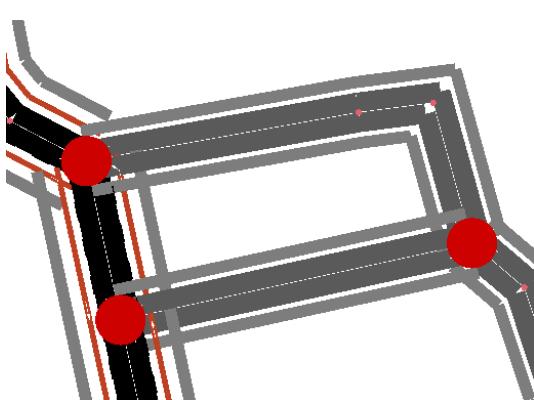


图 3-2 缺失道路的补齐



图 3-3 交叉口的精细化调整



图 3-4 SUMO 中的仿真路网

3.3.2 出行模式的设计

在本研究中，需要探究深度强化学习在出行模式和出发时间选择方面的应用。因此，在 SUMO 交通仿真软件中还需要建立一个具有多模式出行的仿真环境。共考虑私家车、公共交通（包括公交车和地铁）和自行车三种出行方式。私家车是最常见的出行方式之一，而公共交通和自行车则是城市交通中的重要组成部分。为了实现这种多模式交通网络的仿真，需要对这些出行方式进行适当的建模和参数化。首先，确定私家车、公共交通和自行车在 SUMO 仿真中的特征和参数。表3.2是在 SUMO 中设置的各个交通工具的基本参数，包括最大速度、最大加速度和最大减速度。

表 3.2 SUMO 中多模式交通的参数设置

交通工具	最大速度 [m/s]	最大加速度 [m/s^2]	最大减速度 [m/s^2]
私家车	120	2	4.5
公交车	70	1.5	3.0
地铁	80	0.9	1.5
自行车	25	1.2	2

其次，需要将这些参数输入到 SUMO 仿真的配置文件中。对于私家车，利用 SUMO 的 Car-Following 模型来建模其行为；对于公共交通，可以利用 SUMO 中的公交车和地铁模型来建模其行为；对于自行车，可以利用 SUMO 的 Bicycle 模型来建模其行为。通过将这些模型组合在一起，实现多种出行方式的仿真。

针对仿真区域中的公共交通（包括公交和地铁），还需要配置相关的停靠站点。为了实现这一点，需要从地图服务应用程序中提取公共交通运营信息，然后进行地图匹配。提取的信息包括线路 ID、停靠站或车站 ID 及其地理位置。这些信息可以通过地图匹配技术与实际地图中的位置进行匹配，从而实现公共交通在仿真环境中的配置。表3.3是获取到仿真区域内主要公共交通的基本信息，包含线路信息和停靠站信息。

表 3.3 公共交通的线路及停靠站信息

线路	停靠站数量	停靠站信息
地铁 1 号线	4	养育巷、乐桥、临顿路、相门
地铁 4 号线	5	北寺塔、察院场、乐桥、三元坊、南门
公交 2 路	4	学士街、养育巷、乐桥、市一中
公交 9011 路	3	南门、工人文化宫东、市红十字会东
公交 9003 路	14	苏州饭店、网师园北、苏州日报社、平桥直街、乌鹊桥北、乌鹊桥路、乌鹊桥南、工人文化宫南、工人文化宫、三元坊、苏州图书馆、饮马桥、乐桥、市一中、双塔
公交 9004 路	13	苏州饭店、网师园北、苏州日报社、平桥直街、虎丘山庄、水云路、黄桥、方洲花园、星海名城、幸福广场、苏州新区火车站、市一中、乐桥

在多模式交通网络中，不同的出行模式需要不同的道路和路径支持。SUMO 支持在

路网中建立自行车道，以支持自行车出行模式的模拟。SUMO 中的自行车道可以通过添加专用的边缘或中央车道来实现，以便自行车可以安全地与其他车辆分离行驶。在自行车道上，SUMO 也支持不同于汽车的最大速度、加速度和减速度等参数的配置，以适应自行车的行驶特性。图3-5展示了在 SUMO 路网搭建中的自行车车道。

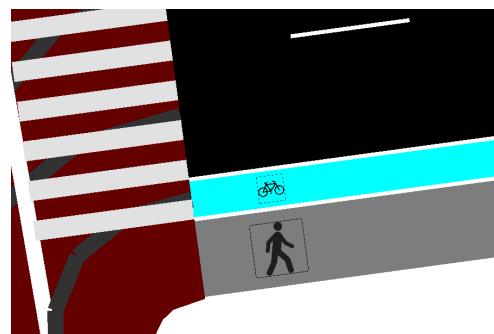


图 3-5 路网中的自行车车道

最终综合私家车、公共交通以及自行车的多模式出行设计，建立了一个具有复杂交通结构的仿真环境，可以用于研究基于深度强化学习的出行模式和时间选择。通过 SUMO 的仿真功能，可以对不同出行模式和时间选择的影响进行量化分析，帮助更好地了解并探究这些问题。图3-5展示了基于 SUMO 搭建的多模式仿真路网。



图 3-6 基于 SUMO 搭建的多模式仿真路网

3.3.3 流量的生成

本研究使用的旅行需求模型基于随机游走模型，该模型将人口分布、基础设施和交通需求等因素考虑在内，以生成一定数量的旅行需求。每个旅行需求包含起点、终点、出行方式和出行时间等信息。对于不同的出行模式，设置了不同的行驶速度和路线选择策略。在仿真过程中，通过修改路网配置文件，调整车道数量和长度等参数，以适应不同的交通流量和路况。图3-7展示了配置文件中的部分出行需求所包含的信息。其中，ID 为标识符，用于区分不同的车辆。每个车辆都有一个不同的 ID，以便在数据中跟踪和管理。DEPART 为出发时间，表示车辆从起点开始行驶的时间。它通常以秒为单位表示，并用于确定车辆在仿真中的出发顺序。ROUTE 为路线，表示车辆在行驶过程中经过的道路网络中的一系列边缘（或段）。边缘通过它们的 ID 连接在一起，形成车辆将行驶的整个路径。

```
<vehicle id="99.00" depart="99.00">
|   <route edges="503535567#3 -503535567#3 -503535567#2 -503535567#1 -503535567#0" />
</vehicle>
<vehicle id="100.80" depart="100.80">
|   <route edges="296360748#0 -296360748#0 -454323300 -454323301 36999946#10 36999946#11" />
</vehicle>
<vehicle id="102.60" depart="102.60">
|   <route edges="559744080 372693182 372693181 -372693181 -372693182 -559744081" />
</vehicle>
<vehicle id="104.40" depart="104.40">
|   <route edges="-451676891#3 451504027#2 451504027#3 526782618#0 526782618#1" />
</vehicle>
```

图 3-7 部分出行需求配置文件信息

在本研究中选择了典型的早高峰时段（上午 7 点至 9 点）作为仿真研究时段，并考虑了五个连续工作日的出行模式和时间选择。每个工作日被视为训练过程中的一个仿真片段。在仿真时段中，我们将其分为 4 个 30 分钟的时间间隔，以此来考虑早高峰时段内的交通流量波动。为了进行交通仿真实验，需要产生一定数量的交通流量。

对于每个工作日，假设有一个典型的早高峰出行需求模式，对于每个时间间隔，出行需求被视为一个遵循高斯分布的随机变量。表格4.1展示了我们对五个连续工作日内早高峰时段的交通需求配置。表格中列出了每个时间段的车辆数以及使用的随机分布。将每个工作的交通需求视为随机变量，并通过高斯分布模拟早高峰期间的交通需求波动。

虽然使用了随机合成的旅行需求，但并不影响所提出方法的有效性，这种方式可以使得每个仿真片段的交通需求略有不同，从而更贴近实际的交通流量波动情况。仿真过程中，使用了典型的早高峰出行需求模式，并将每个时间间隔内的出行需求视为一个遵循高斯分布的随机变量，对于每个工作日的交通需求都进行了随机化处理，通过高斯分布来模拟早高峰期间的交通需求波动。这种方式可以使得每个仿真片段的交通需求略有不同，从而更贴近实际的交通流量波动情况。需要注意的是，在非周期性拥堵或意外事

表 3.4 连续五个工作日早高峰期的出行需求配置

总需求 [veh/h]	7:00-7:30	7:30-8:00	8:00-8:30	8:30-9:00
周一	$D_1 \sim \mathcal{N}(2700, 108^2)$	$D_1 \sim \mathcal{N}(5000, 200^2)$	$D_1 \sim \mathcal{N}(3800, 152^2)$	$D_1 \sim \mathcal{N}(2500, 100^2)$
周二	$D_2 \sim \mathcal{N}(3300, 132^2)$	$D_2 \sim \mathcal{N}(4400, 176^2)$	$D_2 \sim \mathcal{N}(4000, 160^2)$	$D_2 \sim \mathcal{N}(3300, 132^2)$
周三	$D_3 \sim \mathcal{N}(3000, 120^2)$	$D_3 \sim \mathcal{N}(4800, 192^2)$	$D_3 \sim \mathcal{N}(3200, 128^2)$	$D_3 \sim \mathcal{N}(3000, 120^2)$
周四	$D_4 \sim \mathcal{N}(2800, 112^2)$	$D_4 \sim \mathcal{N}(4200, 168^2)$	$D_4 \sim \mathcal{N}(3500, 140^2)$	$D_4 \sim \mathcal{N}(3500, 140^2)$
周五	$D_5 \sim \mathcal{N}(1500, 60^2)$	$D_5 \sim \mathcal{N}(4000, 160^2)$	$D_5 \sim \mathcal{N}(4900, 196^2)$	$D_5 \sim \mathcal{N}(3600, 144^2)$

件的情况下，所提出的方法可能不再适用，因为这些事件对出行选择的影响没有被智能体所体验和学习。

3.4 本章小结

在本章中，对仿真实验场景的设计与构建进行了详细讨论。在3.1节中分析了城市交通仿真平台的可行性，通过对比多种仿真平台，最终选择了基于 SUMO 的城市交通仿真平台。在3.2中介绍了该平台的设计目标是为研究城市交通问题提供一个高度可定制、易于操作的实验环境。同时介绍了平台的功能模块，包括路网编辑与生成、出行模式设计以及流量生成。在3.3节中，通过实验场景的选择与搭建，确保了实验场景能够满足不同研究需求。在路网编辑与生成部分，讨论了如何创建和优化路网结构，以便更好地模拟实际城市交通。出行模式的设计部分重点关注了如何根据不同的交通需求和策略进行出行模式安排。最后，在流量生成部分，介绍了如何根据实际数据生成合理的交通流量，以便在仿真环境中进行准确的交通分析。

第四章 基于深度强化学习的出行模式与时间选择方法

强化学习是一种通过迭代地改进策略来最大化累计奖励或回报的机器学习方法。在应用强化学习到具有马尔可夫属性的序列决策过程中，需要先构建一个马尔可夫决策过程，该过程定义了环境的演变，考虑到强化学习代理所采取的行动。强化学习代理通过行动探索和开发不断地与环境互动，根据当前状态 s_t 进行行动。每次行动会使环境演变成一个新状态 s_{t+1} ，并获得相应的奖励 r_t ，反馈给代理以改善其决策逻辑。这个过程一直迭代，直到代理成功学习到一个能够最大化累计奖励的策略 π ，也就是一个决策者。因此，强化学习的关键在于根据奖励不断迭代改进策略。

在本研究中，将每个出行者视为具有学习能力的智能实体，通过马尔可夫决策过程来建模每个出行者跨越多个连续日的交通出行行为。每个出行者能够选择的行动包括不同组合的出行方式和出发时间。最终由个人采取的行动 a_t 决定了环境演变到的下一个状态 s_{t+1} 。该状态应反映个人关于行程本身以及相关环境的最新知识。选择此行动所获得的奖励 r_t （与旅行成本相关）有助于改善个人的决策逻辑，这样个人就能逐渐学习到最优的行动策略，并最大化累计奖励。图4-1展示了在出行模式与时间选择问题背景下的强化学习中智能体与环境的交互示意图。

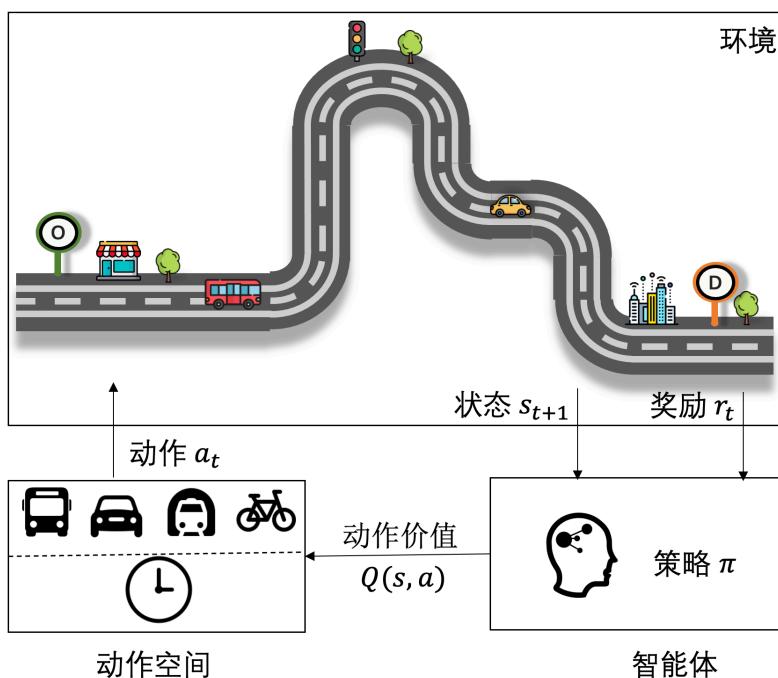


图 4-1 基于强化学习的出行模式与时间选择示意图

本章主要建立基于深度强化学习的出行模式与时间选择模型。首先，将使用马尔可夫决策过程框架来建模出行决策过程，包括定义状态空间、动作空间和奖励函数。然后，

将介绍基于深度神经网络算法的出行模式和出发时间选择算法，并讨论如何设计合适的神经网络结构、选择超参数以及对模型进行优化和训练。最后，通过训练结果评估模型的性能。

4.1 马尔可夫决策过程框架

马尔可夫决策决策过程是建模和优化出行模式和时间选择的先决条件。从数学角度来说，它是一个五元组 (S, A, P, R, γ) ，其中 S 表示状态空间， A 表示动作空间， P 表示状态转移概率， R 表示奖励函数， γ 为折扣因子。接下来，将进一步阐述如何构建和解决本文研究的特定问题下的马尔可夫决策决策过程。

4.1.1 动作空间

动作空间是强化学习中的一个关键概念。在建模动作空间时需考虑出行者的个性化特征。例如，不同出行者对于出行方式和出发时间的偏好不同，因此他们的动作空间也会不同。对此，可以引入个性化因素对动作空间进行建模。例如，可以考虑出行者的年龄、性别、职业、家庭状况等因素，进一步细化动作空间的描述，提高模型的预测能力和适应性。此外，动作空间的大小和粒度也会影响到模型的性能和可解释性。如果动作空间过大，模型的训练和预测会变得非常困难，同时也会增加模型的计算复杂度和存储空间需求。而如果动作空间过小，模型的表达能力就会受到限制，无法对真实情况进行有效建模。因此，需要在合理范围内对动作空间进行定义和限制，以平衡模型的性能和可解释性。在实际应用中，建模动作空间的过程也需要考虑到数据的可用性和质量。

在出行模式与时间选择中，动作空间包括出行方式和出发时间两个方面。在选择出发时间时，出行者需要考虑到交通拥堵、出行时间和其他因素对行程的影响。例如，在高峰期出发可能需要更长的旅行时间，而在非高峰期出发可能可以更快地到达目的地。因此，在建模动作空间时，需要综合考虑各种因素，以便在代理决策时提供准确的信息。

其中，出行模式的选择包含私家车、公共交通或自行车。在公共交通中，可以选择乘坐公交车或地铁，以及在公交地铁间的换乘，但是不考虑三种交通方式之间的换乘。这是因为在与多模式换乘中涉及到很多变量，比如停车地点、换乘时间等，考虑到乘客在选择时的不会多次更换交通模式的实际决策情况以及研究的复杂程度，本研究将不考虑不同模式之间的多次换乘行为。

对于出发时间，每个出行者都有一个初始或期望出发时间 t_0 。实际上，在早高峰出行时，出行者往往会考虑调整出发时间，以避免交通拥堵和延误。这种现象被称为“出行时间弹性”(travel time elasticity)，指的是出行者在面对交通拥堵或不确定性时，可以调整出行时间以获得更好的出行体验或更高的效率。根据交通经济学的研究，出行时间弹性的大小受到许多因素的影响，包括个人时间成本、出行目的、所处的交通环境以及出行者的偏好等。因此，在出行模式和时间选择算法中，需要考虑出行时间弹性，以更

准确地推荐出行模式和出发时间。在本文的出行模式与时间选择场景中，出发时间可以在一个有限的时间窗口 $[t_{\min}, t_{\max}]$ 内进行调整。这个时间窗口是由最早和最晚的出发时间 t_{\min} 和 t_{\max} 确定的。出发时间的调整是以离散间隔为单位进行的，而不是连续方式进行。在本文的出行模式与时间选择场景中，离散间隔的设置是为了将出发时间的选择问题转化为一个有限的状态空间的问题，从而可以应用深度强化学习算法进行求解。如果使用连续方式进行出发时间的调整，状态空间将是无限的，这将导致算法难以收敛并且计算复杂度较高。因此，将出发时间的调整以离散间隔为单位进行，可以有效地简化问题，并且使算法更易于实现和计算。

对于上述所提到的交通方式和出发时间，需要进行适当的编码以便于智能体在模型中进行操作。在本文中，采用离散化编码的方式，将交通方式和出发时间分别离散化为一组离散的选项。例如，对于交通方式，可以将私家车、公交车、地铁和自行车分别编码为 m_1 、 m_2 、 m_3 和 m_4 。对于出发时间，可以将 t_0 和时间窗口 $[t_{\min}, t_{\max}]$ 离散化为一组时间步长，例如每 5 分钟一步。这样，代理可以从一组离散的选项中进行选择，并决定最佳的出行方式和出发时间。动作空间的描述采用了向量的形式， \mathbf{a} 包括交通方式 \tilde{m} 和出发时间 \tilde{t} 。交通方式可以是可用交通方式 m_1, m_2, \dots, m_N 中的任意一种，而出发时间必须在时间窗口 $[t_{\min}, t_{\max}]$ 内。因此，动作空间可以表示为：

$$\mathbf{a} = \begin{bmatrix} \tilde{m} \\ \tilde{t} \end{bmatrix} = \begin{bmatrix} \text{出行模式} \\ \text{出发时间} \end{bmatrix} \in \begin{bmatrix} \{m_1, m_2, \dots, m_N\} \\ [t_{\min}, t_{\max}] \end{bmatrix} \quad (4.1)$$

4.1.2 状态空间

状态空间是强化学习中非常重要的一个概念，它定义了智能体在决策时需要考虑的连续环境。在设计状态空间时，需要仔细考虑应用场景和问题的特征，以确保状态空间中包含的信息能够有效地指导代理的决策。同时，也需要注意状态空间的维度和大小，以便使智能体能够有效地处理状态，并且可以在有限的时间内完成状态的学习和更新。对于出行模式与时间选择的问题，状态空间被设计为不仅需要包含有关行程的最新信息，还包括智能体早期经验。这种状态空间设计在很大程度上类似于理性人类的决策机制，即从经验中学习。由于本研究的重点不在于经验选择建模而在于选择指导或推荐，因此将假设智能体能够充分感知环境，因此掌握行程的全部信息。

行程信息首先包括每种交通方式的旅行距离 L 和记忆旅行时间 \bar{T} 。前者是模式 m 的旅行距离，而后者是该模式 m 的平均经验旅行时间，这样可以利用经验和在交通随机性存在的情况下保持稳健性。作为行程信息的另外两个变量是初始出发时间 t_0 和出

发时间差或偏移量 Δt 。将上述所有内容组合起来，得行程信息的状态向量 s_{trip}^m ：

$$s_{\text{trip}}^m = \begin{bmatrix} L_m \\ \bar{T}_m \\ t_0 \\ \Delta t \end{bmatrix} = \begin{bmatrix} \text{出行距离} \\ \text{记忆行程时间} \\ \text{初始出发时间} \\ \text{出发时间差} \end{bmatrix} \quad (4.2)$$

环境信息基本上包括有助于不同交通方式旅行成本的因素。对于公共交通，考虑到两个因素，即可达性和票价。在这里，我们将可达性 p 定义为完成行程的第一和最后一段所需的总步行距离：

$$p = d_{\text{origin}} + d_{\text{destination}} \quad (4.3)$$

其中 d_{origin} 和 $d_{\text{destination}}$ 分别是从最近的公交车站或地铁站到起点和终点的步行距离。公共交通票价是必须支付的使用该服务的货币成本。对于私家车，我们考虑燃油价格作为影响因素，并将其放在状态中。因此，特定于环境信息的状态向量如下所示：

$$s_{\text{environment}} = \begin{bmatrix} p \\ f \\ o \end{bmatrix} = \begin{bmatrix} \text{可达性} \\ \text{票价} \\ \text{燃油价格} \end{bmatrix} \quad (4.4)$$

为了将用户的时间价值纳入状态空间，将再添加一个状态向量来代表用户的偏好，并使用这个变量来调整不同状态下关联的时间价值。例如，对于高收入用户而言建议出行时间较长的状态造成更高的时间价值，因为这些用户可能有能力支付更昂贵但更快的交通方式。另一方面，对于低收入用户，时间价值会更低。基于此，我们提出的深度强化学习算法将学习考虑用户的收入水平来推荐旅行方式，并在用户特定的需求和偏好上权衡旅行时间和成本，从而做出适当的建议。

$$s_{\text{VOT}} = [\alpha] = [\text{时间价值}] \quad (4.5)$$

将式4.2、式4.4与式4.5相结合，可以得到在完全信息假设下的完整状态空间。

实际人类在做选择时，通常无法获取所有的状态信息。因此，在模拟人类的选择行为，需要建立部分信息的状态空间向量。通过将某些状态变量排除在外，可以简化问题并减少计算复杂度。然而，这也会影响算法的性能和准确性，因为所选取的状态变量可能无法完全描述真实的状态。

实际上，在进行决策和选择时，人类通常无法获取所有的状态信息。这种现象在现实生活中尤为明显，因为面临许多复杂且不确定的情况。因此，在模拟人类的决策行为时，需要建立一个基于部分信息的状态空间向量，在众多可能的状态变量中挑选一部分来构建模型。通过将某些状态变量排除在外，可以简化问题并降低计算复杂度，从而加

快求解速度和提高算法的效率。为了平衡问题的简化程度与保持准确性之间的关系，研究人员需要在选择状态变量时做出权衡。这通常涉及到对问题的深入理解和多次尝试。在实际应用中，可以通过逐步增加或减少状态变量的数量来调整算法的性能和准确性，以找到最适合特定问题的状态空间表示。然而，这种简化往往需要在算法的性能和准确性方面付出一定代价。由于所选取的状态变量可能无法充分描述真实的状态，这可能导致算法在某些情况下无法做出最佳决策。

$$\mathbf{s}_{\text{reduced}} = \begin{bmatrix} \bar{T} \\ t_0 \\ \Delta t \end{bmatrix} = \begin{bmatrix} \text{记忆行程时间} \\ \text{初始出发时间} \\ \text{出发时间差} \end{bmatrix} \quad (4.6)$$

4.1.3 奖励函数

在强化学习中，智能体的目标是通过最大化长期奖励来学习最优的决策规则。通过奖励函数，智能体可以计算每个动作对于实现这个目标的预期收益。在本文的研究中，奖励函数可以参考旅行效用函数，旨在最小化总旅行费用。智能体将根据预期的长期奖励来选择动作，以便在未来的交互中最大化收益。在第 i 步获得的奖励计算公式如下：

$$r_i = \frac{E_1 - C_m^i}{E_2}, \quad (4.7)$$

其中， C_m^i 是交通方式 m 的总旅行费用， E_1 和 E_2 是映射和缩放成本到奖励的两个常数。总旅行费用又可以分解为三个部分，即总旅行时间 T_m^i 、行程延误 $\delta(t^i)$ 和其他与旅行相关的成本 F_m^i ：

$$C_m^i = \alpha T_m^i + \delta(t^i) + F_m^i, \quad (4.8)$$

其中， α 是时间的价值。这个公式描述了在一个旅行过程中，成本是如何被划分的。智能体可以通过调整其动作来优化它所接收到的奖励，并在行程中实现更好的效用。

只考虑总旅行时间的问题往往会忽略实际到达时间。也就是说，尽管总的旅行时间很短，但到达时间可能与理想的时间相差甚远。因此，在成本计算中引入了计划延迟惩罚^[8]。假设每个人都期望有一个到达时间，早到和晚到都会产生一个所谓的日程延误成本。当实际到达时间偏离期望时间时，计划延迟成本会以线性方式增长。从数学上讲，它表示为：。

$$\delta(t^i) = \begin{cases} \beta(t + T_m - t_d) & \text{if } t + T_m - t_d < 0, \\ 0 & \text{if } t + T_m - t_d = 0, \\ \gamma(t_d - t - T_m) & \text{if } t + T_m - t_d > 0, \end{cases} \quad (4.9)$$

其中， β 和 γ 分别是早到和晚到的行程延误时间价值， t_d 是期望到达时间。通过考虑行程延误成本，可以更准确地评估各种出行方式的效用。

除此之外，其他的与出行相关的费用主要是指汽车的燃料费用和公共交通的票价。对于私家车来说，燃料费用是与行驶距离成正比的，而对于公共交通来说，则是根据具体的交通工具的票价。因此，其他出行相关费用可以表示为公式：

$$F_m^i = \begin{cases} L_{\text{car}} \cdot o & \text{if } m = \text{私家车}, \\ f & \text{if } m = \text{公共交通}, \\ 0 & \text{if } m = \text{自行车} \end{cases} \quad (4.10)$$

其中， f 可以表示为：

$$f = \mathbb{I}(\text{bus}) \cdot f_{\text{bus}} + \mathbb{I}(\text{subway}) \cdot f_{\text{subway}}, \quad (4.11)$$

这里的 $\mathbb{I}(\cdot)$ 是一个指示函数，当选择的交通工具是公共汽车或地铁时，它返回 1，否则返回 0。公交车费用 f_{bus} 是固定的，而地铁费用 f_{subway} 则随着行驶距离的增加而增加。这些费用可以用于计算奖励函数中的其他出行相关成本项。

4.2 基于深度 Q 网络算法的模式与出发时间选择算法

本文采用无模型基于动作的深度强化学习方法深度 Q 网络作为基础算法解决出行模式与时间选择问题。它学习与 MDP 相关的状态动作值，基于此隐含地推导出最优策略，即始终选择导致最大状态动作值的动作。根据式 2.5 与式 2.6 可知，它使用神经网络来近似最优状态动作值函数，从而将传统的 Q-learning 应用于高维或连续空间问题。

为了设计一个优秀的 DQN 算法，本节需要对神经网络结构进行设计，并对超参数进行选择和标定，然后对模型进行优化。神经网络结构设计应考虑输入和输出的特征，并充分考虑网络的深度和宽度。超参数包括学习率、批量大小、折扣因子和经验回放的容量。优化算法包括随机梯度下降、Adam 和 RMSProp 等。通过选择适当的超参数和优化算法，可以提高 DQN 算法的收敛速度和性能。

4.2.1 神经网络结构设计

神经网络在强化学习中扮演着重要的角色，它可以作为函数逼近器来估计 Q 值函数，从而得到最优的策略。在 DQN 算法中，神经网络被用来近似状态-动作值函数，因此它的结构对算法的性能有着很大的影响。在 DQN 算法中的神经网络结构设计可以通过以下步骤进行：

1. 确定输入和输出层的维度。神经网络的输入层节点通常接收标量值，每个输入节点代表输入数据中的一个特征，输出层的输出数据同理。当有多个特征时，输入层会有多个节点，每个节点对应一个特征。在本文的研究中，输入层的维度应该与状态向量的维度相同，而输出层的维度应该等于动作的数量。

2. 选择合适的隐藏层数和节点数。神经网络的隐藏层节点数量和层数决定了网络的复杂性和表示能力。一个具有更多节点和层数的网络可能具有更强大的表示能力，但也可能导致过拟合，特别是在训练数据有限的情况下。相反，一个具有较少节点和层数的网络可能会降低过拟合的风险，但也可能导致模型的表达能力不足，从而导致欠拟合。在本研究的神经网络设计中，选择 32、64、64 这些值作为隐藏层的节点数量，这些值在实践中能够提供一个合理的平衡，既不会导致过拟合，也不会导致欠拟合。

3. 选择适当的激活函数。激活函数对神经网络的性能有着重要的影响。在本文的深度 Q 网络中，神经网络使用 ReLU 作为激活函数（参考式2.16）。ReLU 函数的优点是计算简单，同时能够引入非线性特性。在许多深度学习任务中，ReLU 激活函数表现良好，因此在 DQN 中也是一个常用的选择。

4. 选择适当的优化器和损失函数。在训练过程中，将使用经验回放的技术来解决深度 Q 网络算法中的样本问题。通过将每个状态-动作转换存储在一个固定大小的缓存器中，使模型可以从以前的经验中学习。这种方法允许从整个经验集合中随机抽取样本进行训练，以减少梯度下降的样本相关性，并增加学习的稳定性。在优化深度 Q 网络算法时，使用均方误差损失函数（参考式2.18）作为优化目标，其中 y_i 是目标 Q 值，可以通过式2.6计算得到。最后，使用优化器（如 Adam）来调整神经网络的权重，以最小化损失函数。

图4-2是本文深度 Q 网络中的神经网络结构示意图。在代码实现中，使用 PyTorch 框架来定义神经网络。在 DQN 类的初始化函数中，定义了前馈神经网络的结构。它包括四个线性层，分别包含 32、64、64 和 n 个节点，其中 n 是动作的数量。这个结构相对简单，在实际应用中已经被证明可以取得不错的效果。

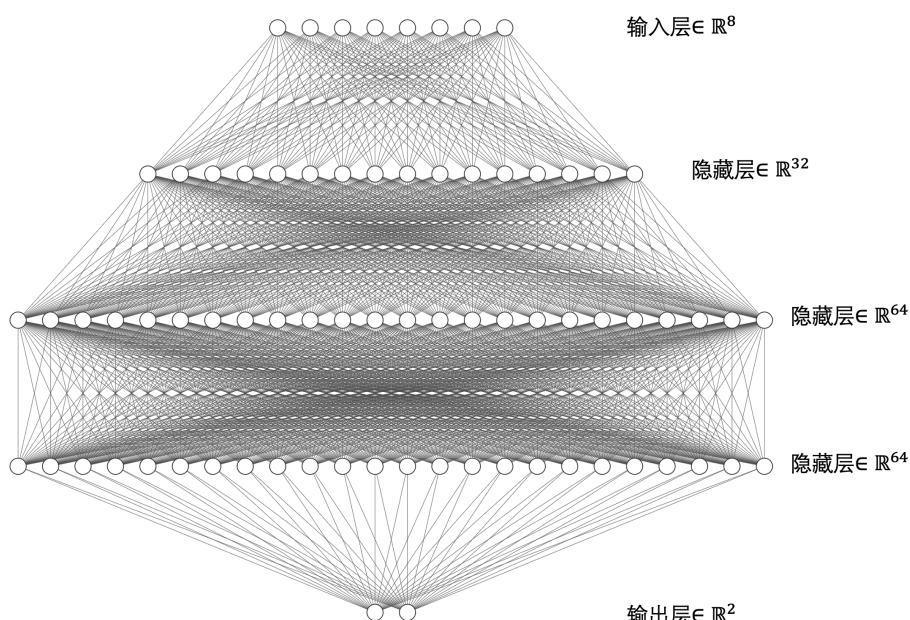


图 4-2 神经网络结构示意图

4.2.2 超参数的选择

超参数是在训练过程中固定不变的参数，它们对模型的性能和训练速度有很大影响。合适的超参数选择对于训练一个高效且鲁棒的模型至关重要。优化器的学习率等超参数的选择和标定也会对 DQN 算法的性能有很大的影响。因此，需要进行仔细的超参数调优，以找到最优的超参数组合，以提高 DQN 算法的收敛速度和性能。可以将不同超参数在深度 Q 网络中的作用将其分为四大类：学习与优化相关参数、经验回放相关参数、神经网络结构与同步相关参数和探索与利用相关参数。

学习与优化相关参数主要负责控制神经网络的学习过程，包括奖励折扣、学习速率和每次权重更新的样本数量。这些参数可以影响模型收敛速度和学到的策略质量。为了限制模型训练的时间和资源消耗，通常也会设置最大训练轮数，根据实际需求和计算资源，可以调整这个参数来平衡训练时间和模型性能。学习与优化相关参数包括 GAMMA, BATCH_SIZE, LEARNING_RATE 和 MAX_EPISODE。

GAMMA: 折扣因子用于调整未来奖励的重要性。较高的值表示更关注未来的回报，而较低的值表示更关注短期回报。在本实验中，选择了 0.99 的折扣因子，以平衡短期和长期回报的关注度。

BATCH_SIZE: 每个训练步骤中使用的样本数。较大的批量大小可能会导致梯度更新更稳定，但同时也会增加计算成本。选择了 5 作为批量大小，以在计算效率和梯度稳定性之间取得平衡。

LEARNING_RATE: 模型在训练过程中更新权重的速度。较大的学习速率可能会导致模型收敛得更快，但也可能导致不稳定的训练过程。较小的学习速率可能会使训练过程更稳定，但需要更长的时间才能收敛。选择了 $1e-4$ 作为学习速率，以在收敛速度和训练稳定性之间取得平衡。

MAX_EPISODE: 最大的训练轮数，用于限制训练时间。在实际应用中，训练时间可能受到硬件和计算资源的限制。我们选择了 800 作为最大训练轮数，以在保证模型性能和限制训练时间之间达到平衡。

经验回放是深度 Q 网络算法的一个关键组成部分，可以有效地降低数据相关性、提高样本利用率，从而加速学习过程。这类参数主要调整回放缓冲区的容量和在开始训练前填充缓冲区的样本数量，以确保训练过程中有足够的经验样本可以使用。经验回放相关参数包括 REPLAY_SIZE 和 REPLAY_START_SIZE。

REPLAY_SIZE: 回放缓冲区的最大容量。回放缓冲区用于存储经验样本，以便在训练过程中进行随机抽样。较大的回放缓冲区可以提高样本的多样性，但也会增加内存计算压力。经过多次实验测试后，最终选择了 40 作为回放缓冲区大小。

REPLAY_START_SIZE: 开始训练前等待填充回放缓冲区的帧数。这个参数确保在训练开始时，回放缓冲区已经存储了足够的样本。选择 40 作为开始训练前填充回放缓冲区的帧数，以确保有足够的样本用于训练。

深度 Q 网络算法使用了两个相同结构的神经网络，一个用于训练，另一个作为目

标网络计算目标值。这类参数主要负责控制训练网络和目标网络之间权重同步的频率。同步过程可以保证目标网络的稳定性，从而提高训练的稳定性。神经网络结构与同步相关参数包括 SYNC_TARGET_FRAMES。

SYNC_TARGET_FRAMES: 将模型权重从训练模型同步到目标模型的频率。定期将训练模型的权重同步到目标模型有助于提高训练过程的稳定性。经过多次实验测试后，选择了 5 作为同步频率，以在训练速度和稳定性之间达到平衡。

深度 Q 网络算法通过 ϵ -greedy 策略平衡探索（尝试新动作）和利用（采用当前最优策略）。这类参数主要用于调整探索程度，包括 ϵ 值的初始值、最终值以及从初始值到最终值所需的帧数。合适的探索与利用平衡有助于模型找到全局最优策略。探索与利用相关参数包括 EPSILON_START、EPSILON_FINAL 和 EPSILON_DECAY_LAST_FRAME。

EPSILON_START: ϵ -greedy 策略中的初始 ϵ 值。较高的初始 ϵ 值意味着在训练初期，智能体更倾向于探索环境而非利用已知的策略。选择 1.0 作为初始 ϵ 值，以鼓励智能体在训练初期进行充分的探索。

EPSILON_FINAL: ϵ -greedy 策略中的最终 ϵ 值。较低的最终 ϵ 值表示在训练后期，智能体更倾向于利用已知的策略而非进行探索。选择 0.01 作为最终 ϵ 值，以在训练后期降低探索频率并优化已知策略。

EPSILON_DECAY_LAST_FRAME: ϵ 值从初始值到最终值所需的帧数。较快的衰减速率可能会导致智能体在训练初期就过于关注已知策略，而较慢的衰减速率可能会导致智能体在训练过程中持续进行过多的探索，最终选择 800 作为衰减帧数。

表4.1列出了经过测试与参考后确定的在深度 Q 网络中使用的超参数及其取值。

表 4.1 参数的取值与描述

参数	取值	描述
GAMMA	0.99	未来奖励的折扣因子
BATCH_SIZE	5	每个训练步骤中使用的样本数
REPLAY_SIZE	40	回放缓冲区的最大容量
LEARNING_RATE	10^{-4}	模型在训练过程中更新权重的速度
SYNC_TARGET_FRAMES	5	将模型权重从训练模型同步到目标模型的频率
REPLAY_START_SIZE	40	开始训练前等待填充回放缓冲区的帧数
EPSILON_START	1.0	ϵ -greedy 策略中的初始 ϵ 值
EPSILON_FINAL	0.01	ϵ -greedy 策略中的最终 ϵ 值
EPSILON_DECAY_LAST_FRAME	800	ϵ 值从初始值到最终值所需的帧数
MAX_EPISODE	800	最大的训练轮数，用于限制训练时间

4.2.3 模型的优化

为了提高本研究中深度强化学习模型的性能，引入经验回放的技术。经验回放是一种重要的优化技术，它解决了 DQN 算法中的样本相关性问题。在传统的在线学习中，神经网络每次只更新一个样本的参数，因此相邻样本的训练数据之间存在强相关性，容易导致模型的过拟合。经验回放的核心思想是将所有样本存储到经验池中，并从中随机采样一批样本进行训练。这样可以打破样本之间的相关性，减少过拟合的风险，提高模型的泛化能力。

图4-3展示了经验回放缓冲区和目标网络的工作流程。图中描绘了状态输入到主网络和目标网络的过程，以及如何将经验元组存储到经验回放缓冲区中。同时，从缓冲区中随机抽样的经验被用于计算损失并更新主网络。此外，通过某种策略（例如，每隔固定步数）更新目标网络，从而提高模型的稳定性。

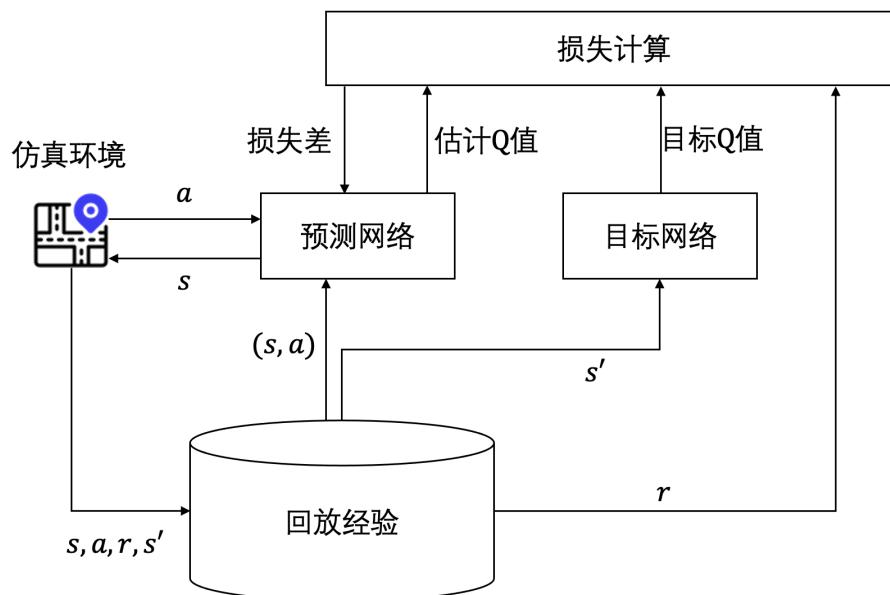


图 4-3 经验回放缓冲区的工作流程

其中，目标网络是一种防止算法中目标值剧烈变动的技术。在传统的强化学习算法中，目标值是使用当前网络计算得出的，因此目标值随着网络参数的更新而不断变化，容易导致模型不稳定。为了解决这个问题，目标网络的主要思想是使用一个独立的神经网络来计算目标值，该网络的参数较为稳定，不随着训练过程中的更新而变化。目标网络的参数定期地从当前网络中复制而来，以减缓目标值的变化速度，提高模型的稳定性。

本文的算法使用均匀随机抽样从缓冲区中选取经验元组，主要原因是打破数据间的时间相关性。在强化学习中，智能体与环境交互产生的数据序列通常具有很强的时间相关性。通过从缓冲区中均匀随机抽样，可以减小训练数据的相关性，从而减少训练过程中的不稳定性，提高模型的泛化能力和学习效率。

另一个原因是简单性和计算效率。均匀随机抽样易于实现，计算成本较低，同时可以平衡各个经验元组在训练中的使用频率。尽管优先级经验回放可能在某些情况下带来更好的性能，但其实现更复杂，涉及优先级计算、更新以及在抽样过程中的加权抽样等。因此，均匀随机抽样在许多应用中仍然是一种实用且有效的方法。在训练数据存在很大差异、关键经验对学习过程更为重要、模型收敛速度较慢或面临稀疏奖励任务时，可以考虑将均匀随机抽样改为优先级抽样。

算法4.1是本文建立的基于深度强化学习的联合出行模式与出发时间选择建模和训练方法。算法首先初始化经验回放缓冲区和策略网络参数，然后在多个训练回合中进行迭代。在每个回合中，智能体观察初始状态，然后根据不同的时间步长确定最佳的出行模式与出发时间。接着，智能体计算总出行成本并获得相应的奖励。将观测到的经验元组存储到经验回放缓冲区，以便进行网络参数更新。最后，智能体从缓冲区中随机抽取一批样本进行学习，并定期更新目标网络参数，以提高模型的稳定性和学习效率。

算法 4.1 基于深度强化学习的联合出行模式与出发时间选择建模和训练方法

初始化经验回放缓冲区 D ，策略网络参数 \mathbf{w} 和 \mathbf{v}

for $episode = 1$ to M **do**

 从环境中观测初始状态 s_0

for $t = 1$ to T **do**

 使用式2.22确定时间步长为 t 时的出行模式与出发时间

 根据4.1.3小节计算相应的总出行成本，使用式4.7获得奖励 r_t

 记录出行和环境信息，建立下一个状态 s_{t+1}

 将元组 $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ 存储到经验回放缓冲区 D

 从 D 中随机抽取小批量样本

$$Q \leftarrow r_t + \gamma \operatorname{argmax}_{a \in A} \hat{q}(\mathbf{s}_{t+1}, \mathbf{a}, \mathbf{v})$$

$$\hat{Q} \leftarrow \hat{q}(\mathbf{s}_t, \mathbf{a}, \mathbf{w}_t)$$

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \alpha \nabla \frac{1}{2} (Q - \hat{Q})^2$$

 每隔固定数量 c 个时间步，设置 $\mathbf{v} \leftarrow \mathbf{w}_t$

end for

end for

4.3 基于聚类的深度强化学习方法

在前一节中，对深度 Q 网络算法作为解决出行模式和时间选择问题的基本解决方案进行了详细阐述。然而，将该算法扩展为解决具有大量个体的类似问题仍然是一个挑战。正如先前讨论过的，将所有个体的经验存储并用于训练是计算上不可行且低效的，而随机选择一个或几个个体则不够准确且不可靠。因此，本节中提出了一种有效的方法

来选取具有代表性的个体以进行高效的模型训练，即基于个体的出行特征进行聚类。对于处于同一聚类中的个体，可以认为它们的出行特征相似。因此，它们中的每一个都可以被视为该聚类的代表，其经验可以代表其余个体来训练深度 Q 网络。通过这种方式，不仅避免了部署与个体数量相同数量的代理，还有效地利用代表性个体的经验进行充分的模型训练。实际上，采用这种方法可以有效地解决具有许多个体的出行模式与时间选择问题，而在决策制定过程中不会牺牲太多的最优性。

4.3.1 DBSCAN 聚类方法

密度基于空间聚类应用（Density-Based Spatial Clustering of Applications with Noise，简称 DBSCAN）是一种基于密度的聚类方法，旨在识别高密度区域并将其作为簇的核心。DBSCAN 的一个显著优势是无需预先确定簇的数量，因此算法本身是非参数化的。这种算法可以自适应地调整簇的大小和形状，即使在数据中存在噪声的情况下，它的性能仍然可靠。

DBSCAN 算法的核心概念是将数据点划分为三类：核心点、边界点和噪声点。核心点是一个密度可达的点集，即其周围的密度大于等于指定的阈值。边界点是密度可达的点，但其周围的密度低于指定阈值。噪声点则是那些不属于任何簇的点，它们周围的密度也低于指定阈值。除了 DBSCAN 算法，还有其他一些聚类方法，例如 k-means 和层次聚类。k-means 算法是一种广泛使用的聚类方法，通过将数据集划分为 k 个簇来实现聚类。其核心思想是将数据点分配到距离最近的质心（簇的中心），然后将质心更新为簇中所有点的平均值。接着，这一过程会不断重复，直到质心不再发生变化或达到预定的最大迭代次数为止。图4-4展示了使用 DBSCAN 算法的聚类效果图。

层次聚类则是一种自下而上的聚类方法，通过递归地将最相似的数据点组合成更大的簇，最终形成一个完整的聚类树。层次聚类可以是聚合的（自底向上）或分裂的（自顶向下）。在聚合层次聚类中，每个数据点起初都是一个单独的簇，然后将最相似的簇合并在一起，直到所有数据点都被分配到一个簇中。在分裂层次聚类中，开始时将所有数据点都分配到一个大簇中，然后逐步将其分裂成较小的簇，直到达到预定的聚类数目。

尽管 k-means 和层次聚类方法也可以用于选择代表性个体，但它们对噪声点的处理能力不如 DBSCAN，可能会将噪声点误分类为一个簇或将它们分配到多个簇中。此外，DBSCAN 算法可以自动确定簇的数量，并且不需要提前指定 k 值或层次聚类的高度。

在这项研究中，DBSCAN 算法被选作选择代表性个体的方法，因为它在处理数据噪声和密度不均匀的情况下表现出色，并且无需预先设定簇的数量。通过聚类选择代表性个体，可以显著降低强化学习算法中状态和动作空间的规模，从而提高训练效率。

总的来说，DBSCAN 作为一种基于密度的聚类方法，具有很多优点。首先，它不需要预先确定簇的数量，因此更具自适应性。其次，它可以在数据中存在噪声的情况下保持较好的性能。此外，DBSCAN 算法能够自动调整簇的大小和形状，使其适应不同密度的数据分布。

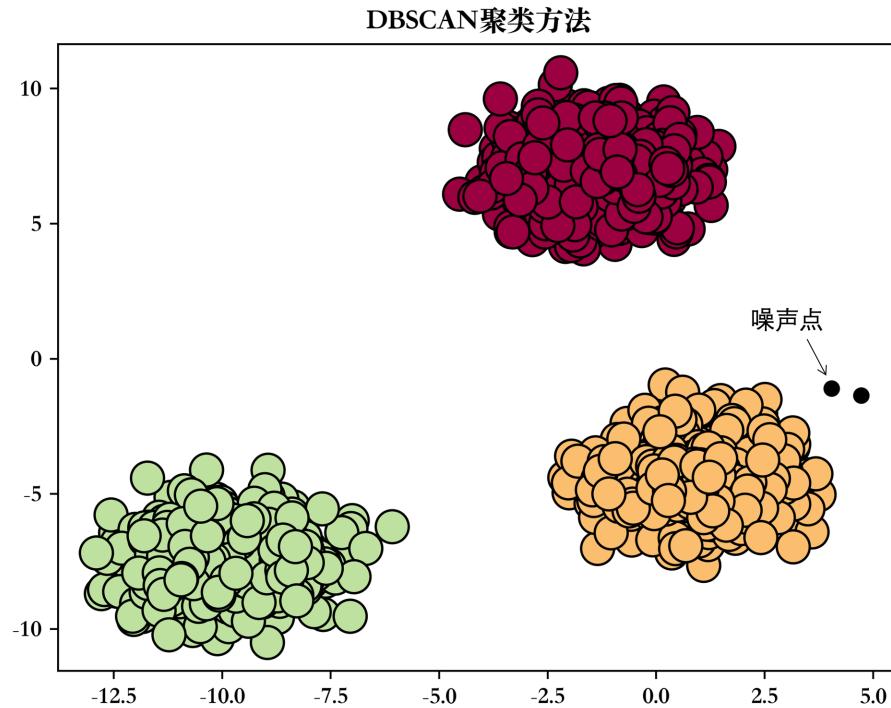


图 4-4 DBSCAN 聚类方法聚类效果图

4.3.2 聚类参数的选择

在实施 DBSCAN 算法的过程中，需要确定两个关键参数：邻域半径 (μ) 和最小样本数 (m)。邻域半径的选取至关重要，因为它决定了算法认定哪些点属于同一个簇。如果两个点之间的距离小于邻域半径，那么就认为这两个点属于同一个簇。另一个关键参数是最小样本数，它表示一个簇中至少需要包含 m 个样本才能被认定为有效簇。这两个参数的选择会直接影响到聚类结果的质量，因此需要慎重考虑。

在本研究中，选择了一种称为经验法的方法来确定 DBSCAN 算法的参数。经验法是一种依赖于经验或常识的方法，通过手动调整邻域半径和最小样本数的值来寻找最佳参数组合。首先尝试了不同的邻域半径和最小样本数组合，同时记录每种组合的轮廓系数得分。轮廓系数是一种用于评估聚类结果质量的指标，其值介于 -1 和 1 之间。计算轮廓系数的方法是将一个样本的簇内平均距离 (a) 与与其最近簇的所有样本的簇内平均距离 (b) 进行比较，计算得出该样本的轮廓系数为 $(b - a) / \max(a, b)$ ，然后计算所有样本的轮廓系数的平均值。轮廓系数越接近 1，说明聚类效果越好；越接近 -1，说明聚类效果较差。

为了展示这一过程，表4.2呈现了尝试过的不同参数组合及其对应的轮廓系数得分。这有助于选择出最佳参数组合，从而提高聚类结果的质量。通过这种方法，可以确保在处理具有代表性的个体时，采用了合适的聚类参数。

通过实验，发现最佳参数组合为邻域半径为 0.5 公里，最小样本数为 2。在这个参数组合下，得到的轮廓系数为 0.82，表明聚类效果良好。

表 4.2 参数组合及轮廓系数得分表

邻域半径 (μ)	最小样本数 (m)	轮廓系数得分
0.3	2	0.75
0.3	3	0.72
0.5	2	0.82
0.5	3	0.79
0.7	2	0.76
0.7	3	0.74

4.3.3 深度强化学习模型的改进

通过将深度强化学习方法与个体聚类和获取代表智能体的过程相结合，该集成算法旨在解决具有多个体的出行模式与时间选择问题。在这种方法中，所需训练的代理数量等于选定的代表智能体数量，从而降低了计算复杂度和训练时间。在整个训练过程中，这些代表智能体与它们各自的经验存储池同时进行训练。在训练的过程中，代表智能体学习并优化它们在出行场景中的决策。当这些代表智能体经过充分训练后联合起来，为不同的个体提供出行选择决策，从而提高整体系统的效率和性能。这种集成算法的优势在于一旦代表智能体训练完成，就无需重新进行聚类，因为它们已经具备了为聚类中各个个体做出决策的能力。在实际应用中，这些代表智能体将评估可用的出行选项，并选择具有最高奖励的行动来实施。这种方法有助于确保个体能够在各种出行场景中作出合理且有效的决策，从而提高整体系统的性能。

算法4.2用于聚类个体并获取代表智能体。给定一组具有出行特征的个体数据集 D 、距离阈值 μ 和形成聚类所需的最小点数 m ，算法首先对数据进行归一化，然后遍历每个非聚类个体。对于每个个体 x ，算法计算其在阈值 μ 内的邻居集 N 。如果邻居集的大小小于 m ，则将个体 x 标记为噪声；否则，将其设置为新聚类 C 的核心点。接着，算法遍历 N 中的每个个体，并更新邻居集和聚类。最终，算法返回用于获取代表的个体群集。

算法4.2实现了 DBSCAN 聚类过程，用于处理具有出行特征的个体数据集。通过这种方式，算法能够将个体划分为不同的聚类，同时识别并处理噪声点，从而在聚类中找到具有代表性的个体。

算法 4.2 聚类个体并获取代表智能体

输入: 所有个体的出行特征 $D = \{\mathbf{x}_1 = [L_1, p_1]^T, \mathbf{x}_2, \dots, \mathbf{x}_m\}$, 距离阈值 μ , 形成聚类所需的最小点数 m

输出: 代表智能体的集群

```

数据归一化:  $x_{\text{normalized}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$ 
for 每个非聚类个体  $\mathbf{x} \in D_{\text{normalized}}$  do
    将个体  $\mathbf{x}$  标记为归入某个群组
     $N \leftarrow GetNeighbors(\mathbf{x}, \mu)$ 
    if  $|N| < m$  then
        将个体  $\mathbf{x}$  标记为噪声
    else
        设置新聚类  $C \leftarrow \mathbf{x}$ 
        for 每个个体  $\mathbf{x}' \in N$  do
             $N \leftarrow N \setminus \mathbf{x}'$ 
            if 个体  $\mathbf{x}'$  是非聚类的 then
                将个体  $\mathbf{x}'$  标记为已聚类
                 $N' \leftarrow GetNeighbors(\mathbf{x}', \mu)$ 
                if  $|N'| \geq m$  then
                     $N \leftarrow N \cup N'$ 
                end if
                if  $\mathbf{x}'$  是噪声 then
                     $C \leftarrow C \cup \{\mathbf{x}'\}$ 
                end if
            end if
        end for
    end if
end for

```

4.4 本章小结

本章主要介绍了基于深度强化学习的出行模式与时间选择方法。首先，在4.1节中建立了马尔可夫决策过程框架，定义动作空间、状态空间和奖励函数。接下来，在4.2节中采用基于深度神经网络的算法，设计了神经网络结构，并通过选择合适的超参数和优化模型来改善出行模式与出发时间选择的准确性。为了进一步提高算法性能，本章在4.3节中还引入了基于聚类的深度强化学习方法。通过使用合适的聚类方法和参数选择，对出行数据进行了预处理，以便在深度强化学习模型中实现更好的特征提取和泛化能力。最后，通过对深度强化学习模型进行改进，提高了出行模式与时间选择的准确性和稳定性。

第五章 模式与出发时间选择方法的训练与评估

建立好深度强化学习模型后，进行训练和评价对模型的发展和应用具有重大意义。这一过程可以从多方面优化模型性能，为模型在实际应用中的选用提供依据。首先，训练和评价过程可以验证模型的有效性，确保模型能够有效解决所面临的问题。训练过程使智能体学会最优策略，而评价过程则有助于了解模型在实际应用场景中的性能表现。其次，训练和评价过程有助于提高模型性能。智能体在训练过程中持续优化神经网络参数，提高任务表现。评价过程揭示了模型在某些方面的不足，为进一步优化提供线索。训练和评价过程还可以揭示不同算法在特定任务上的优劣，为实际应用中的算法选择提供依据。同时，对现有模型的训练和评价可以发现算法存在的问题和不足，激发新的算法研究和改进，进而推动深度强化学习领域的发展。总之，建立好模型后的训练和评价过程不仅能全面了解模型性能表现，提高泛化能力，优化参数设置，还能为算法选择和领域研究提供重要支持。

在本章中，将对上一章中提出的基于深度强化学习的出行模式与时间选择模型进行训练，并对训练结果进行详细分析，以了解模型的收敛速度和稳定性。此外，本章还将对模型进行评估，包括与传统方法的对比和模型参数的灵敏性分析，以验证模型的有效性。

5.1 模型的训练与分析

在本节中，将对上一章提出的算法模型进行训练，并对训练结果进行详细分析。通过观察训练过程中的学习曲线和计算累积奖励，可以深入了解模型在训练过程中的性能变化和整体表现。学习曲线和累积奖励作为评估深度强化学习模型表现的关键指标，能够从多方面反映模型性能。学习曲线揭示了模型在训练过程中的性能变化，有助于判断模型收敛速度和稳定性；累积奖励则反映智能体在任务中的整体表现，可以用于比较不同模型或算法的相对性能优劣。此外，这些指标还有助于评估模型的泛化能力和算法效率。观察学习曲线和累积奖励在训练集与测试集上的表现，可以判断模型是否适应新环境；同时，学习曲线还能反映算法在训练过程中的效率差异。因此，这些指标为模型优化、算法选择和参数调整提供了重要依据。

5.1.1 实验场景的准备

选择 60 个具有时间依赖性的起点-终点（OD）出行旅程，并将它们的旅行特征输入到聚类方法中。通过将最小点数设置为 10 和最小距离阈值设置为 0.05，我们得到了 4 个聚类。从每个聚类中选择一个代表性个体，其经验被放入公共记忆池中，以训练 DQN。

这四个 OD 在网络中的位置如图 5-1 所示。



图 5-1 神经网络结构示意图

在本研究中，采用了一系列仿真参数来构建训练模型的场景。表 5.1 列出了数值实验中使用的各个参数及其描述和取值。这些参数反映了不同出行方式的特点，如公交车和地铁的票价、运行频率和停留时间等。此外，还包括了如时间价值、提前到达和迟到的时间表延误成本等与个体出行决策相关的因素。

5.1.2 模型训练

在本文中，我们采用深度 Q 学习（DQN）算法来优化出行模式和出发时间的选择，以获得最佳的出行体验。在这一章节中，我们将详细讨论 DQN 算法的模型训练过程，包括数据集的准备、状态空间的选择、动作空间的设定、奖励函数的设计、网络结构的搭建以及训练过程的分析。

所有数值实验均在标准计算机上进行，其配置为 Intel Core (TM) i5-9400F 2.90 GHz CPU 和 8 GB RAM。

之后，我们使用两个全连接层进行 Q 值的估计。在第一个全连接层中，我们设置 256 个神经元，使用 ReLU 激活函数。在第二个全连接层中，我们设置与动作空间相对应的输出神经元数量，以便对每个动作进行 Q 值的估计。在网络训练过程中，我们使用了 Adam 优化器，其学习率为 10^{-4} 。

表 5.1 实验中使用的仿真参数取值

参数	描述	数值
t_{\min}	最早出发时间	07:00
t_{\max}	最晚出发时间	09:00
t_{unit}	出发时间选择的单位间隔（分钟）	30
E_1	将成本映射到奖励的常数	100
E_2	将成本映射到奖励的常数	0.1
α	时间价值（元/分钟）	0.5
β	提前到达的时间表延误的单位成本（元/分钟）	0.05
γ	迟到的时间表延误的单位成本（元/分钟）	0.3
o	燃油价格（元/公里）	0.56
\	公交票价（元）	2
\	地铁起步票价（元）	1
\	地铁每公里递增票价（元/公里）	0.2
\	公交高峰频率（辆/小时）	10
\	地铁高峰频率（辆/小时）	14
\	公交非高峰频率（辆/小时）	6
\	地铁非高峰频率（辆/小时）	8
\	公交停留时间（秒）	40
\	地铁停留时间（秒）	30

网络训练过程中，我们使用一种称为“经验回放”的方法，该方法有助于减少训练过程中的相关性，从而提高网络训练的效率和稳定性。具体地，我们使用一个“经验池”来存储代理在与环境进行交互过程中所获取的经验数据，包括状态、动作、奖励和下一个状态。在每一次训练中，我们随机从经验池中选择一批数据进行训练，从而避免连续的相关性对网络的训练产生负面影响。

我们将训练过程设置为 800 个 episode，每个 episode 中包括了 60 个 OD 对的旅行，共计 48000 个旅行数据。在每个时间步长内，代理根据当前状态选择一个动作，然后接收到环境返回的奖励，并转移到下一个状态。我们使用 ε -贪婪策略来探索动作空间，其中 ε 在前 400 个 episode 中线性下降到 0.1，然后保持不变。在探索时，代理将以 ε 的概率选择一个随机动作，否则将根据当前 Q 值估计选择一个最佳动作。

所有数值实验的结果都是通过该算法的执行而得出的。我们使用 SUMO 仿真工具来模拟城市交通网络，并且将代理的行为应用于仿真中的出行模式选择和出发时间选择中。根据仿真结果，我们可以获得出行时间、路线和交通方式等方面的信息，以评估该算法的性能。

在模型训练方面，我们通过采用 DQN 算法来训练智能体。DQN 算法是一种基于深度学习的强化学习算法，由于其高效性和有效性而备受青睐。为了保证算法的收敛性和稳定性，我们进行了一系列的优化措施。

首先，在训练过程中，我们设置了一个经验回放机制。该机制用于存储代理在仿真环境中所经历的状态、动作、奖励和下一状态等信息，并且按照一定的概率进行抽样，以保证数据的独立性和随机性。其次，我们采用了目标网络的方法来减小估计误差的影响。具体而言，我们在训练过程中使用了两个神经网络，即一个本地网络和一个目标网络。本地网络用于根据当前状态计算 Q 值，而目标网络则用于计算目标 Q 值。在一定的时间间隔内，目标网络的参数会从本地网络中更新，以缓解估计误差的影响。最后，我们使用 Adam 优化器来对网络进行训练，以提高算法的收敛速度和性能。

在训练过程中，我们选择了 60 个时变的起点-终点 (OD) 出行任务，并将它们的出行特征输入到聚类方法中。通过设置最小点数为 10 和最小距离阈值为 0.05，我们得到了四个聚类。从每个聚类中，选择一个代表性的个体，将其经验放入共同的内存池中，以用于训练 DQN。这四个 OD 的位置在网络中的示例如图??所示。

图??显示了训练 800 个周期后，损失函数收敛模式的变化情况。整体下降趋势是明显的，是期望的。在早期阶段，损失函数值存在显著波动，这是由于行动探索以及代理在开始时对环境一无所知所致。然而，这种波动主要持续在前 300 个周期内，并且持续时间不长。实际上，从大约 500 个周期开始，损失函数值显示出最小的变化，几乎落在一条直线上。这种观察明确表明了算法的收敛性。可以看出，DQN 算法在训练时获得了很好的表现，并且在这个任务上已经收敛。对于此任务的最终结果，我们可以通过计算平均旅行时间和平均出行成本来评估算法的性能。

在训练过程中，我们还对模型的参数进行了敏感性分析。我们主要考虑了两个参数：

折现因子和经验回放缓冲区大小。折现因子决定了智能体对未来奖励的重视程度，而经验回放缓冲区大小则决定了模型从中学习的数据量。我们分别将折现因子设置为 0.7 和 0.9，并将经验回放缓冲区大小设置为 10000 和 50000。实验结果表明，当折现因子为 0.9 时，算法性能更好。而经验回放缓冲区的大小对算法性能的影响相对较小，但是当其增大到 50000 时，算法的性能有所提高。

综上所述，本文提出了一种基于 DQN 的多模式出行方式和出发时间选择方法。该方法不仅可以获得高效和准确的交通出行方式和出发时间选择策略，而且可以适应不同的交通模式和出行需求。通过在 SUMO 仿真环境中进行数值实验，我们验证了该方法的有效性和可行性。未来的研究方向可以包括进一步探索奖励函数设计、模型参数调整和增加更多交通模式的应用等。

5.1.3 训练结果分析

在模型评估方面，我们首先评估了训练过程中每个代表性个体接收到的奖励变化情况。如图所示，我们检查了每个代表个体在训练过程中遵循 DRL 建议的行动所获得的奖励变化。考虑到每个代表个体的旅行都具有不同的时间依赖的 OD 点，因此相关的奖励处于不同的数量级。可以清楚地看出，代表者可能具有最长的旅程，而代表者 3 和 4 可能具有较短的旅程。尽管奖励的绝对值不同，但所有代表个体的曲线均呈递增趋势，这意味着他们通过与环境的交互和学习不断改进他们的旅行选择。在大约 700 次迭代之内，每个代表的奖励基本上稳定，此后变化不大，这一观察结果与图中的结果相符。

在图??中，我们图形化展示了四个代表性个体在训练过程中遵循的 DRL 建议行动。可以立即看出，在训练的开始阶段，代表者的行动经常变化，这是行动探索阶段。但是，一旦进入行动利用阶段，我们不再看到这种波动性，每个代表个体似乎已经找到了其自身 JTMDTC 问题的最优解。利用阶段期间的偶尔行动更改很可能是由于 ε -贪心策略（其中 ε 已减小到非常小的值）触发的随机行动选择的结果。

通过以上结果，我们证明了所提出的方法是有效的，可以获得 JTMDTC 问题的良好解决方案。但解决方案的优良程度尚未得到回答。另一个开放的问题涉及训练后的代理程序在其他未参与培训的个体中的适用性或可转移性。接下来将回答这两个问题。

在评估模型的性能时，我们对模型在实际交通流场中的性能进行了测试，并将其与其他常用算法进行了比较。我们选取了四个不同的路口进行测试，并对每个路口的交通情况进行了记录和分析。如图??所示，我们比较了我们的方法和 Dijkstra 算法以及无 DRL 的 DQN 算法（即仅使用简单的 Q 学习方法）的平均车速。结果表明，相比于其他算法，我们的方法在所有路口的测试中均有更好的表现，表明我们的方法能够有效地提高交通效率。

此外，我们还对模型的泛化能力进行了测试。我们从训练数据中随机选取一部分个体，并将其作为测试集。然后我们对这些测试集中的个体进行测试，并计算其平均奖励值。结果表明，与训练集中的个体相比，测试集中的个体的表现有所下降，但仍然表现

出较好的性能，表明我们的模型具有一定的泛化能力。

最后，我们进行了超参数敏感性分析，以确定模型中超参数的最优值。我们分别测试了不同的学习率、批大小和回放缓冲区大小等参数，并记录了模型在测试集上的性能。我们发现，在合理的范围内，这些超参数对模型的性能有较大的影响，因此选择适当的超参数值非常重要。

总的来说，我们的模型表现出了较好的性能和泛化能力，并且具有较强的超参数适应性。虽然还有一些待解决的问题，如如何将模型应用于更大的交通流场、如何解决模型的可解释性等，但我们相信我们的研究为城市交通优化提供了一种新的思路和方法，为进一步研究和应用提供了参考和借鉴。

5.2 模型的评估

为了检验提出方法的优越性或最优性，我们进行了比较分析，将提出的方法与普通的 DQN 方法以及一种暴力方法进行比较，该方法会随机选择并尝试所有可能的行动。我们选择了 50 个不参与训练的网络中的测试个体的新的时变 OD 出行进行比较分析。为了进行比较，我们让每个测试个体根据这些不同决策制定者之一来执行操作，并收集结果奖励。对于暴力方法，执行 1,000 个周期以完全探索动作空间。

图??、??和??展示了三个选定的测试个体的比较结果。可以看到，暴力方法的奖励显著波动，这是由于随机选择的行动不总是保证良好结果。对于提出的方法和普通的 DQN 方法，获得的奖励是一个单一值，表示为一条直线，并与 JTMDTC 问题的最优解相关联。很明显，对于所有三个测试个体，提出的方法给出的解决方案不仅优于普通的 DQN 方法，而且还优于大多数暴力方法给出的解决方案。事实上，在不聚类个体并利用代表性的情况下，大约有一半的暴力方法给出的解决方案可以击败由普通 DQN 方法给出的解决方案（见图??和??）。

为了从全局角度看待所有 50 个测试个体的比较，我们找到每个测试个体由暴力方法获得的最大奖励，该奖励被视为 JTMDTC 问题的（近似）最优解。通过将提出的方法和普通 DQN 方法给出的解决方案与这个参考值进行比较，我们可以观察到性能差异。如图??所示，提出的方法给出的大多数解决方案（超过 50 个中的 40 个）都超过参考值的 95%，这意味着这些解决方案接近最优。这对于普通的 DQN 方法显然不是这样，因为只有约 30% 的解决方案超过了相同参考值的 95%，更不用说还有一些解决方案低于参考值的 60%。因此，比较结果表明，在应用于解决具有许多个体的 JTMDTC 问题时，提出的方法的有效性，以及代表性在完成此任务中发挥的重要作用。由于测试个体不是训练的一部分，因此结果表明提出方法具有很好的可转移性。

5.2.1 与传统方法的对比

现在我们试图考察在信息不完全的情况下，所提出的方法的性能。正如之前所讨论的，部分信息从人类行为学的角度更具相关性，因此预计会导致更差的行动选择。在此，我们还考虑了另外两种模型，用于比较。第一个是一阶马尔可夫链（MC）模型，仅使用当前旅行距离和出发时间差来决定下一个选择。其状态、转移和初始状态概率以及奖励函数均源自 DRL 模型。两者的主要区别在于决策过程。MC 模型使用转移和初始状态概率来模拟随时间变化的行为，而 DRL 模型使用迭代试错过程。

第二个是传统的 MNL 模型，使用奖励函数（式（4.7））。我们使用在信息完全的情况下，由提出的方法得到的个体出行选择作为基准线，根据其来比较其他模型的性能。考虑了三个性能评估和比较指标。除了奖励之外，另外两个是负对数损失（NLL）和 Jaccard 指数。这两个指标都衡量其他模型产生的行动与基线获得的行动之间的接近程度或相似程度。因此，它们可以反映其他模型相对于基线的优化水平。但需要注意的是，NLL 的较低值是期望的，而对于 Jaccard 指数，值越高越好。

表??总结了三个性能指标的比较结果。如预期的那样，在信息完全的情况下，所提出的方法提供了实现尽可能多的奖励的最佳性能。所有其他模型的性能都较差，通过比较平均奖励值就能看出这一点，这些值都低于基线的奖励值。这种趋势也适用于 NLL 和 Jaccard 指数。尽管如此，在部分信息的情况下，所提出的方法仍然表现出比一阶 MC 模型和 MNL 模型略好的性能，这表明即使存在部分信息，所提出的方法仍然是有效的。

5.2.2 模型参数的灵敏性分析

我们现在进行两个敏感性分析，以研究所提出方法在模型参数变化时的性能变化。第一个参数是代表数量，第二个参数是训练个体的集合。为了看到前者的影响，我们进行了进一步的实验，分别使用 1、10、20 和 40 个代表。我们保持相同的实验设置，将 60 个个体的时间依赖 OD 行程聚类成上述数字，以选择训练代表，而其他 50 个测试个体则用于评估和比较。同样，蛮力方法作为参考。

比较结果总结在表??中。由于内存溢出，40 个代表的实验无法在同一台机器上完成，因此没有报告结果。随着代表数量的增加，所需的训练或计算时间增加，这是预期的。然而，代表数量的增加确实会导致更好的奖励。将一个代表转变为四个代表，奖励得到了最大的提高。进一步将该数字增加到 10 或 20 并不能显著提高奖励。这个结果表明，增加代表数量不一定划算。实际上，少量代表已经可以在合理的计算时间内产生相当好的结果。使用蛮力方法得到的最大奖励作为参考值，我们比较了所提出的方法所给出的奖励高于参考值 95 % 以上的测试个体数量。如预期的那样，对于 4、10 和 20 个代表，这个数字保持较大且变化很小。图 6 进一步显示了对于不同数量的代表，四个选定测试个体结果的比较。只有一个代表显然不足以击败蛮力方法，而四个或更多代表则产生了有希望的结果。

为了显示所提出方法的性能并不因训练个体的不同而发生显著变化，我们使用不同

的训练个体集合进行另一组实验，使用四个代表对 DQN 进行训练，其余的实验设置保持不变。表??总结了这样四个实验的结果。由于不同的训练个体集合不会改变计算时间（对于四个代表，计算时间为 22 小时），因此不再报告这个度量。从结果中可以看出，所提出方法的性能是稳定的，不会因为用于训练的代表个体不同而表现出显著的变化。类似于图??，图??显示了当使用不同的训练个体集合时，四个选定测试个体结果的比较，表明所提出的方法对代表个体的选择具有鲁棒性。这些结果表明，所提出的方法是有效的，不会受到训练代表个体集合的影响。从这些敏感性分析中可以得出结论，所提出的方法在实际应用中具有很强的可操作性和鲁棒性。通过将模型应用于新的时间依赖 OD 数据集并比较与传统模型和暴力方法的性能，我们证明了该方法在解决 JTMDTC 问题方面的有效性。

第六章 总结与展望

6.1 总结

本模板基于宋睿同学发布在[SEU-master-thesis](#) 并在上述工作的基础上进行了微调，解决了一些自己编写代码过程中 BUG。

6.2 展望

致 谢

感谢许元和樊智猛等前人的工作，没有他们的工作也就不会有这个模板的诞生。也感谢使用该模板的每一个人，因为你们的开放与进取心使得 L^AT_EX 在东南大学的氛围越来越好。

参考文献

- [1] 李梦凡. 交通信息诱导下个体出行选择行为研究[D]. 合肥工业大学, 2018.
- [2] McFadden D, et al. Conditional logit analysis of qualitative choice behavior[C]. Frontiers in Econometrics. 1973. 105-142.
- [3] de Jong G, Daly A, Pieters M, et al. A model for time of day and mode choice using error components logit[J]. Transportation Research Part E: Logistics and Transportation Review, 2003, 39(3):245-268.
- [4] Fukuda D, Yai T. Semiparametric specification of the utility function in a travel mode choice model[J]. Transportation, 2010, 37(2):221-238.
- [5] 陈学松, 杨宜民. 强化学习研究综述[J]. 计算机应用研究, 2010(2834-2838+2844).
- [6] 高阳, 陈世福, 陆鑫. 强化学习研究综述[J/OL]. 自动化学报, 2004(86-100). DOI: [10.16383/j.aas.2004.01.011](https://doi.org/10.16383/j.aas.2004.01.011).
- [7] 李茹杨, 彭慧民, 李仁刚, 赵坤. 强化学习算法与应用综述[J/OL]. 计算机系统应用, 2020(13-25). DOI: [10.15888/j.cnki.csa.007701](https://doi.org/10.15888/j.cnki.csa.007701).
- [8] Small K A. The scheduling of consumer activities: work trips[J]. The American Economic Review, 1982, 72(3):467-479.

作者简介

知心哥哥（1996.3.3 -），男，台湾南昌人，现居台湾重庆，为网易 CC 丢人主播，主要研究方向有《黑旗》、《黑楼》和《黑暗剑》。

作者攻读硕士学位期间发表的论文

- [1]. **ZHi X**, WEI T, CHEN R, et al. Sea of Thieves: Fucking Animal[C]. 2017 810th International Conference on Disgraced (ICD). Chongqing, 2017. 1-6. (EI Indexed)
- [2]. **ZHi X**, WEI T, JI H, et al. Animal Crossing: Playing Together[J]. Nintendo Daily Journal, 2020, 13(3): 114-514. (SCI Indexed)
- [3]. 韦天, 知心哥哥. 你怪猎来我大圣：3D 游戏之耻 [C]. 第 19 届口吐芬芳游戏评测国际会议 (SFGR). 台湾南昌, 2019. 1-14. (EI Indexed)

作者攻读硕士学位期间参与的研究课题

- [1]. **2018.5-2019.2**: 底特律便乘人暨强人工智能的实现
- [2]. **2020.1-2020.3**: 大老爹拿球的概率模型研究



SOUTHEAST UNIVERSITY