



廣東工業大學  
GUANGDONG UNIVERSITY OF TECHNOLOGY

# 硕士学位论文

(专业学位)

基于强化学习的电影推荐系统的研究

作者姓名：李翔宇

导师姓名：陈玮

学科(专业)或领域名称：控制工程

论文答辩年月：2021年5月



分类号：

学校代码：11845

UDC：

密级：

学 号：2111804327

## 广东工业大学硕士学位论文

（工程硕士）

# 基于强化学习的电影推荐系统的研究

李翔宇

导师姓名（职称）：陈玮（教授）

曹志广（研究员）

学科（专业）或领域名称：控制工程

学 生 所 属 学 院：自动化学院

答 辩 委 员 会 主 席：李迪（教授）

论 文 答 辩 日 期：2021 年 5 月 24 日

A Dissertation Submitted to Guangdong University of Technology  
for the Degree's Master of Engineering

# Research on movie recommendation system based on Reinforcement Learning

Candidate: Xiangyu Li  
Supervisor: Prof. Wei Chen

May 2021  
School of Automation  
Guangdong University of Technology  
Guangzhou, Guangdong, P. R. China, 510006

## 摘要

推荐系统出身于互联网时代,用于解决大规模信息爆发问题,可以有效帮助人们在数字大海中寻找需要的信息或商品。

目前,互联网公司大多都为他们的系统配备了推荐引擎来取得更好的收益,比如美国著名的电子商务网站亚马逊,其收入的百分之 30 得益于在线系统部署的推荐引擎。在国内,同样有一大批公司在其在线系统上使用着推荐技术。例如淘宝、京东、拼多多等人们经常使用的在线购物服务商;网易云、爱奇艺等娱乐服务供应商等,正是这些在线服务的存在使得我们生活方便快捷又多姿多彩。

目前现有的推荐系统大多都是基于统计学知识以及机器学习方法构建,通过对用户的历史记录进行处理,来获得用户画像和物品画像,随后将这些特征输入各种推荐系统模型进行运算从而产生一个的结果。然而这些推荐方式仍属于静态推荐,因为他对于用户的反馈信息不能动态地加以应用,所以推荐的结果只能是针对单一时刻的效果而无法做到长效动态地进行推荐。为了实现长效动态的推荐效果,本文决定将强化学习与推荐系统进行结合。强化学习是一种用于实现序列交互过程收益最大化的学习方法,很适合解决目前推荐系统存在的问题。

本文针对传统的推荐算法只能静态周期性更新而缺乏长效性这一问题,提出了一种基于强化学习的电影推荐系统,系统采用离线收集的用户交互日志中的数据来对推荐算法进行训练,通过对日志数据进行处理来获取可以用于强化学习推荐算法的训练数据,同时对在强化学习推荐算法训练过程中产生误差的来源进行分析,并提出了解决方法。主要工作内容和结论如下:

(1) 设计了一种基于强化学习的电影推荐系统,同时提出了一种日志数据的处理方法,用于生成强化学习推荐算法训练所需要的强化学习四元组数据。

(2) 针对经典的强化学习算法进行修改,分别实现了基于 DQN 的推荐算法、基于 DDPG 的推荐算法以及基于 Reinforce 的推荐算法。分析了以 Movielens-100k 数据集进行训练时,强化学习推荐算法出现误差的来源。

(3) 针对使用离线数据集训练强化学习推荐算法容易出现训练误差的问题,提出一种将模仿学习与强化学习相结合的推荐算法,通过在训练过程中将状态与动作进行

约束来限制强化学习产生误差。

(4) 针对模仿学习与强化学习在同一个网络训练时, 存在训练效果不稳定不可控的问题, 本文提出一种解耦网络的改进训练方法, 通过使用两个网络分别训练强化学习和模仿学习, 并使用一个参数来调节模仿学习对于强化学习动作选择范围的限定程度, 以此来控制训练效果的偏向。

(5) 通过在全新数据集上再次实验来测试本文提出的推荐系统的效果, 最终验证了本文所提出的推荐系统以及改进训练方法的有效性。

**关键词:** 强化学习; 推荐系统; 模仿学习

## ABSTRACT

Recommender system originated from the Internet age, which is used to deal with the problem of large-scale information explosion. It can help people search the information or goods they need in the digital sea.

At present, most Internet companies have equipped their systems with recommendation engines to obtain better returns. For example, the famous US e-commerce website Amazon, 30% of its revenue is due to the recommendation engine deployed by the online system. In China, there are also a large number of companies using recommended technologies on their online systems. For example, Taobao, JD, Pinduoduo and other online shopping service providers that people often use; NetEase Cloud, iQiyi and other entertainment service providers, etc., it is the existence of these online services that make our current life convenient, fast and colorful.

The great majority of the existing recommendation systems are using statistical and machine learning. They obtain user portraits and item portraits by processing the user's historical records, and then input these features into various recommendation system models for calculations to generate a Judging the results. However these recommendation methods are still a static recommendation, because it cannot dynamically apply user feedback information. Therefore, the result of the recommendation can only be aimed at the effect of a single moment and cannot be a long-term dynamic recommendation. In order to achieve a long-term dynamic recommendation effect, this paper decided to combine reinforcement learning with a recommendation system. Reinforcement learning is a learning method used to maximize the benefits of the sequence interaction process, and it is very suitable for solving the problems of the current recommendation system.

Aiming at the problem that the traditional recommendation algorithm can only be updated periodically and lacks long-term effectiveness, this paper proposes a movie recommendation system based on reinforcement learning. The system uses offline collected user interaction log data to train the recommendation algorithm. , By processing the log data

to obtain training data that can be used for the reinforcement learning recommendation algorithm, at the same time, it analyzes the sources of errors in the reinforcement learning recommendation algorithm training process, and proposes an improved method. The main work content and conclusions are as follows:

(1) I design a film recommendation system based on reinforcement learning, and a log data processing method is proposed to generate the reinforcement learning quadruple data required for the training of reinforcement learning recommendation algorithm.

(2) The classic reinforcement learning algorithm is modified, and the recommendation algorithm based on DQN, the recommendation algorithm based on DDPG, and the recommendation algorithm based on Reinforce are implemented respectively. The sources of errors in the reinforcement learning recommendation algorithm are analyzed when the Movielens-100k data set is used for training.

(3) Aiming at the problem that the use of offline data sets to train reinforcement learning recommendation algorithms is prone to training errors, a recommendation algorithm that combines imitation learning and reinforcement learning is proposed, which restricts reinforcement learning by constraining states and actions during the training process Produce errors.

(4) Aiming at the problem of unstable and uncontrollable training effects when imitation learning and reinforcement learning are trained on the same network, this paper proposes an improved method for decoupling networks, which uses two networks to train reinforcement learning and imitation learning separately. And use a parameter to adjust the degree of limitation of imitation learning for the selection range of reinforcement learning actions, so as to control the bias of the training effect.

(5) Test the effect of the recommender system by experimenting on a new data set, and finally verify the effectiveness of the recommender system and improved method proposed.

**Key words:** Reinforcement learning; Recommendation system; Imitation learning

# 目录

摘要.....	I
ABSTRACT.....	III
目录.....	V
CONTENTS.....	VIII
<b>第一章 绪论.....</b>	<b>1</b>
1.1 研究背景及研究意义.....	1
1.2 国内外研究现状.....	3
1.3 本文研究目标和主要研究内容.....	5
1.4 本文结构安排.....	6
<b>第二章 推荐方法概述.....</b>	<b>7</b>
2.1 协同过滤方法.....	7
2.1.1 基于内存的协同过滤技术.....	7
2.1.2 基于模型的协同过滤技术.....	9
2.1.3 混合协同过滤技术.....	10
2.2 深度学习方法.....	10
2.2.1 AutoRec.....	11
2.2.2 递归神经网络.....	12
2.2.3 GraphSAGE.....	13
2.3 本章小结.....	13
<b>第三章 强化学习概念与理论介绍.....</b>	<b>14</b>
3.1 强化学习概念.....	14
3.2 强化学习方法.....	16
3.2.1 基于值的方法.....	16
3.2.2 深度决策网络 DQN.....	18
3.2.3 基于策略的方法.....	19
3.2.4 Actor-Critic 算法.....	20



3.3 本章小结.....	21
<b>第四章 特征处理与模型设计改进.....</b>	<b>22</b>
4.1 系统框架设计.....	22
4.2 特征嵌入表示.....	22
4.2.1 数据集选取.....	22
4.2.2 Word2vec 训练.....	23
4.3 基于强化学习的推荐算法.....	25
4.3.1 基于 DQN 的推荐算法.....	25
4.3.2 基于 Policy Gradient 的推荐算法.....	26
4.3.3 基于 DDPG 的推荐算法.....	27
4.4 强化学习推荐算法的性能改进.....	29
4.4.1 基于 DDPG 的推荐算法的失效分析.....	29
4.4.2 强化学习推荐算法误差来源分析.....	30
4.4.3 DQN 推荐算法改进方法.....	31
4.4.4 解耦网络改进方法.....	31
4.5 本章小结.....	33
<b>第五章 实验环境设置与结果分析.....</b>	<b>34</b>
5.1 数据集生成及评价指标.....	34
5.1.1 训练数据生成.....	34
5.1.2 评价指标.....	35
5.2 实验环境.....	36
5.3 强化学习推荐算法对比实验.....	36
5.3.1 有效性对比实验.....	36
5.3.2 性能对比实验.....	37
5.3.3 改进强化学习推荐算法对比实验.....	39
5.3.4 解耦网络效果对比实验.....	40
5.4 系统测试.....	41
5.5 本章小结.....	44

---

---

结论与展望.....	45
参考文献.....	47
攻读学位期间取得与学位论文相关的成果.....	52
学位论文独创性声明.....	53
致谢.....	54

# CONTENTS

<b>ABSTRACT(IN CHINESE).....</b>	<b>I</b>
<b>ABSTRACT(IN ENGLISH).....</b>	<b>III</b>
<b>CONTENTS(IN CHINESE).....</b>	<b>V</b>
<b>CONTENTS(IN ENGLISH).....</b>	<b>VIII</b>
<b>Chapter 1 Introduction.....</b>	<b>1</b>
1.1 Background and significance of research.....	1
1.2 Research status at home and abroad.....	3
1.3 Research objectives and main research contents of this article.....	5
1.4 Structure of this paper.....	6
<b>Chapter 2 Overview of Recommendation Methods.....</b>	<b>7</b>
2.1 Collaborative filtering method.....	7
2.1.1 Memory based collaborative filtering technology.....	7
2.1.2 Model-based collaborative filtering technology.....	9
2.1.3 Hybrid collaborative filtering technology.....	10
2.2 Deep learning methods.....	10
2.2.1 AutoRec.....	11
2.2.2 Recursive Neural Network.....	12
2.2.3 GraphSAGE.....	13
2.3 Chapter summary.....	13
<b>Chapter 3 Concepts and theories of reinforcement learning are introduced.....</b>	<b>14</b>
3.1 Concept of reinforcement learning.....	14
3.2 Reinforcement learning methods.....	16
3.2.1 Value-based approach.....	16
3.2.2 Deep decision network DQN.....	18
3.2.3 Policy based approach.....	19
3.2.4 Actor-Critic algorithm.....	20

3.3 Chapter summary.....	21
<b>Chapter 4 Feature Processing and Model Design Improvement.....</b>	<b>22</b>
4.1 System Frame Design.....	22
4.2 Feature Embedding Representation.....	22
4.2.1 Data set selection.....	22
4.2.2 Word2vec training.....	23
4.3 Recommendation algorithm based on reinforcement learning.....	25
4.3.1 Recommendation algorithm based on DQN.....	25
4.3.2 Recommendation algorithm based on Policy Gradient.....	26
4.3.3 Recommendation algorithm based on DDPG.....	27
4.4 Performance improvement of reinforcement learning recommendation algorithm..	29
4.4.1 Failure analysis of recommendation algorithm based on DDPG.....	29
4.4.2 Analysis of error in reinforcement learning recommendation algorithm.....	30
4.4.3 DQN recommendation algorithm improvement method.....	31
4.4.4 Decoupling network improvement method.....	31
4.5 Chapter summary.....	33
<b>Chapter 5 Experimental Environment Setting and Results Analysis.....</b>	<b>34</b>
5.1 Data set generation and evaluation indicators.....	34
5.1.1 Training data generation.....	34
5.1.2 Evaluation Index.....	35
5.2 Experimental environment.....	36
5.3 Reinforcement learning recommendation algorithm comparison experiment.....	36
5.3.1 Effectiveness comparison experiment.....	36
5.3.2 Performance comparison experiment.....	37
5.3.3 Improved reinforcement learning recommendation algorithm comparison experiment.....	39
5.3.4 Decoupling network effect comparison experiment.....	40

5.4 Test of the system .....	41
5.5 Chapter summary.....	44
<b>Conclusion and prospect.....</b>	<b>45</b>
<b>References.....</b>	<b>47</b>
<b>Publication and patents during study.....</b>	<b>52</b>
<b>Statement of original authorship and copyright licensing declaration.....</b>	<b>53</b>
<b>Acknowledgements.....</b>	<b>54</b>

# 第一章 绪论

## 1.1 研究背景及研究意义

近几年来,随着信息技术的进步和网上服务的普及,人们能够迅速获得大量信息。在互联网飞速发展的今天,得益于数据存储能力的提升以及计算机算力的飞速增长,一大批互联网企业也如雨后春笋般冒出,互联网+的模式几乎在出现在了各行各业,引起了巨大的社会变革。即使是最普通的用户,也可以立即使用互联网企业的产品查询几乎所有有关该产品相关信息,例如服务功能的描述、附带的广告、其他用户的相关评论和评分等。虽然获取信息是一种有价值的能力,但人们面临着大量的数据源,使他们无法找到有用和适合的内容,从而导致出现信息过载问题<sup>[1]</sup>。如何帮助用户对这些信息进行筛选,找出最适合该用户的一部分产品成为众多互联网企业面临的一大难题。

推荐系统作为一种为企业发现商品售卖规律,从而产生额外利润的捆绑销售方法,在大数据时代,也找到了其独特的意义<sup>[2][3]</sup>。它可以通过向用户提供个性化的内容以起到过滤掉其他无用的信息的效果,是一种有效的信息过滤工具,同时还能辅助用户做出某些决策。大多数的互联网用户在日常活动中会利用到这些服务,比如阅读书籍、观看视频、听音乐或者购物等一些活动,这些服务也提高了用户的日常生活体验。另一方面,企业在提供服务时,由于用户和物品存在长尾现象,即少数的用户为活跃用户以及少部分商品为火爆商品,所以浪费了很大一部分服务资源,而推荐系统可以为用户推荐长尾部分适合他的产品,推广了大部分的商品并提高了商品的覆盖率和多样性<sup>[4][5][6]</sup>。

目前,互联网公司大多都为他们的系统配备了推荐引擎来取得更好的收益,比如美国著名的电子商务网站亚马逊,其收入的百分之 30 得益于在线系统部署的推荐引擎;相似的还有著名的电影电视服务供应商奈菲公司,它的电影推荐系统几乎为网站提供了全部的用户流量<sup>[7]</sup>。在国内,同样有一大批公司在其在线系统上使用着推荐技术。在电子商务领域,有淘宝、京东、拼多多等人们经常使用的在线购物服务商;在影音娱乐领域,有网易云、爱奇艺等娱乐服务供应商。正是这些在线服务的存在使得我们现在的的生活方便快捷又多姿多彩。

电影推荐是众多个性化服务里面的一个分支。近几年来,电影行业的发展迅速蓬勃,

以及观影方式的增加，网络观影这种碎片化时间的娱乐方式得到了更多互联网用户的青睐，而一般的电影视频网站，大多都存有成百上千部的电影，每一部电影同时包含多个标签，例如影片的类型，电影的主演以及导演等。如果不使用推荐系统来为人们进行个性化推荐，单凭电影分类标签等引导用户搜寻自己想看的电影，效率是十分低下的。构建有效的电影推荐系统可以解决此问题<sup>[8][9]</sup>。

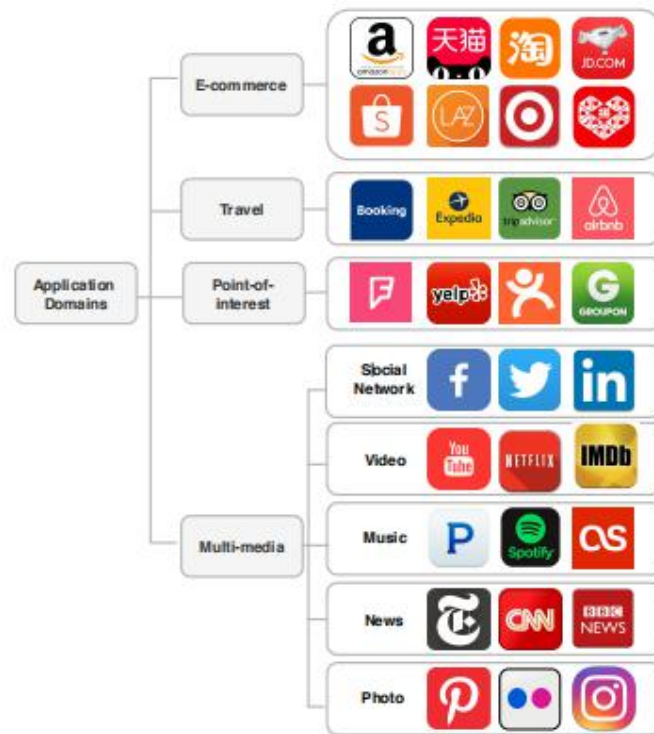


图 1-1 带有推荐系统的各类服务

Fig.1-1 Various services with a recommendation system

广义下的推荐系统包括数据和推荐算法两部分。在不同的应用场景下，获取数据后，选择不同的推荐算法来挖掘数据的特征并进行推荐。若是推荐系统视为一个黑盒，那么系统输入的就是推荐算法所需格式的训练数据，输出的则是推荐结果<sup>[10][11]</sup>。一般用于推荐的数据包含以下几种不同信息：

(1) 用户/物品信息：包括用户的年龄、性别、职业、爱好等个性化信息，以及物品的类型、价格、用处等功能类别信息。

(2) 上下文信息：包括用户与推荐系统发生会话时的时间、地点等信息，个别还包括周边标志性建筑类场景信息。

(3) 显示反馈信息：包括用户对物品的评分、收藏或购买等能够直接表示用户兴趣的行为信息，不过此类数据一般较为稀少。

(4) 隐式反馈信息：包括用户的点击、浏览等隐式交互信息，相比于显示反馈信息更容易获取，处理得当也能起到较好的推荐效果。

然而的推荐系统大多都是基于统计学知识以及机器学习方法，通过对用户的历史记录进行处理，来获得用户画像和物品画像，随后将这些特征输入各种推荐系统模型进行运算从而产生一个的评判结果。这种推荐方式仍属于静态推荐，因为他对于用户的反馈信息不能动态地加以应用。因此本文试图引入强化学习来解决传统的推荐算法只能静态周期性更新而缺乏长效性这一问题。

强化学习用于在交互过程中寻求最优策略来收获最多的奖励或者实现某个目的<sup>[21]</sup>。因为强化学习的模型是马尔可夫决策模型<sup>[23]</sup>，而推荐系统与用户交互的过程也是一种类似的决策过程，所以如何将强化学习应用于推荐系统也是目前众多互联网公司所在探索研究的重要方向。

## 1.2 国内外研究现状

推荐系统最早是由 P.Resnick 等人在 1997 年左右提出，用于解决信息量爆发问题，近年来也是一个较为火热的研究方向<sup>[12]</sup>。目前来说，推荐系统可以被认为是一种以机器学习为基础的个性化决策推理系统。其通过分析用户与物品的交互历史以及用户与物品各自的相关信息来帮助用户找到其最可能需要的物品。

在推荐系统的传统领域里，主要有协同过滤推荐，基于内容的推荐以及两者融合的混合推荐三种方法，存在的问题主要有数据稀疏与冷启动问题<sup>[13]</sup>。早在 28 年前，研究人员就设计了第一个基于协同过滤的推荐系统，只不过当时并未使用推荐系统这个概念<sup>[14]</sup>。该小组创建了 MoviesLens 网站并将它作为一个研究推荐方面知识的重要平台，该网站的数据集也是目前学术界推荐领域引用量最大的数据集。协同过滤算法针对数据量稀少的问题可以获得不错的效果，其工作原理就是通过统计获得用户或者物品之间的相似性，再以此为推荐依据来为用户做出推荐，Sarwar 等人在本世纪初第一次使用了协同过滤算法<sup>[15][16]</sup>。

2003 年亚马逊公司发表一篇文章，完整详细的阐述了如何将协同过滤推荐算法与亚马逊平台相结合<sup>[17][18]</sup>。在 2006 年，美国影视服务供应商奈菲，第一次举办了推荐系



统方向的比赛，为推动推荐系统的发展做出了巨大的贡献，该比赛利用已有的用户行为历史记录，对电影进行评分<sup>[19]</sup>。在此次赛事中矩阵分解法被提出，此后几年内，矩阵分解成为推荐系统领域的重要方法之一，其中最常见的方向是利用隐语义模型，通过隐含特征的方式，将用户与物品联系到一起<sup>[20]</sup>。

SimonFunk 于 2006 年提出著名的 Funk-SVD 算法，对用户-物品矩阵进行矩阵分解操作后获得两个较低阶的特征矩阵，并通过误差最小化获得原矩阵缺失部分。此处的误差通常采用均方根误差来衡量，Salakhutdinov 等于 2008 年对已有的矩阵分解方法进行改进，提出了经典的 PMF 概率矩阵分解模型，通过假设用户与物品分别独立，将用户物品的隐含特征带入模型进行梯度下降训练，来获得用户与物品的打分情况<sup>[19]</sup>。

矩阵分解模型主要是利用用户的显示反馈来解决数据稀疏问题，然而当数据量过大时，用户-物品评分矩阵将会变得十分占用内存，计算起来也更加缓慢，所以研究人员转而研究混合推荐模型。

Liu 等提出的推荐模型能够充分学习用户的兴趣分布，从而提高推荐效果<sup>[24]</sup>。Song Qiang 等于 2015 年，提出增量矩阵分解模型，该模型可以应用于在线推荐系统，能够有效的应对数据更新问题，在用户与物品的交互记录出现更新时能够迅速对用户物品进行特征更新<sup>[25]</sup>。

2012 年，Jiang 等人提出了一种社交推荐模型，研究表面具有好友关系的用户往往兴趣爱好也相似，所以也可以利用社交信息进行推荐<sup>[26][27]</sup>。2018 年，Zhou 等在利用用户-物品评分记录的同时，加上用户社交关系，学习用户兴趣偏好，并根据用户兴趣偏好进行推荐<sup>[28]</sup>。

进入到深度学习时代，得益于深度学习强大的特征提取和学习能力，将深度学习与推荐系统相结合也成为推荐系统新的热门研究方向<sup>[29]</sup>。当前，基于深度学习的推荐方法以及被成功应用到了实践中，其主要是利用深度学习强大的数据处理能力，将特征学习与推荐集成为一个整体。一般基于深度学习的推荐系统主要有输入、模型和输出三部分构成，其中输入的还是一般的用户与物品的信息，模型部分则包含大部分的深度学习模型，输出得到的一般是用户或物品的特征表示，或者进一步对特征做处理获取推荐目标<sup>[30]</sup>。深度学习促进推荐系统的发展主要体现在：

(1) 特征表示：通过深度学习的 Embedding 技术，能够使得数据集的信息被充分

挖掘，用户与物品的特征得以被扩展，由此也大大加强了数据集的表达能力，便于推荐系统对数据进行识别推荐。

(2) 特征交互：由于深度学习的网络都是有很多层组成，所以特征被选取之后也可以获得深层次的交叉，免去了传统推荐系统需要研究人员进行手动选取特征组合的缺点，而特征交叉对于数据的表达能力又能够进一步提升，从而达到提升模型的推荐效果。

Sedhain 等于 2015 年提出了 AutoRec 模型，此模型以自编码器为框架，使用梯度下降法训练<sup>[31]</sup>。AutoRec 模型首次成功将深度学习技术应用于推荐系统。

2016 年，YouTube 团队提出了基于多层感知机（MLP）的推荐模型，此模型能够有效利用数据信息，通过将这些信息输入到模型来获取用户的表示，然后选取与用户最相似的几个用户，将他们的视频推荐给用户，从而过滤掉大量视频<sup>[32]</sup>。

阿里巴巴团队在 2018 年提出了深度兴趣网络推荐模型，此模型在输入阶段，使用固定长度的用户、物品和候选广告特征向量，使用 ActivationUnit 分别计算物品权重，最后对物品向量进行加权求和，与用户特征向量以及其他特征进行 concat 处理，最后输入 DNN 网络进行预测<sup>[53]</sup>。

综上所述发现，不论是传统的推荐算法还是使用深度学习的推荐算法，他们所利用的手段，都可以概括为获取数据的特征，然后对用户进行推荐，而对于数据的交互过程本身却没有进行利用，如何将交互过程结合强化学习应用于推荐系统，以及是否能带来有效的提升是目前推荐系统领域研究一个新的方向<sup>[33][34]</sup>。

### 1.3 本文研究目标和主要研究内容

本文的研究目标是设计出一种可以通过数据集进行训练的基于强化学习的电影推荐系统框架，将深度强化学习技术应用于电影推荐方面。同时还提出一种将模仿学习与强化学习结合的方法，用于解决强化学习系统通过日志文件进行训练时会出现的误差问题。主要研究工作和创新点如下：

(1) 提出一种基于强化学习的电影推荐系统框架的设计方法，同时提出了一种日志数据的处理方法，用于生成强化学习推荐算法训练所需要的强化学习四元组数据

(2) 针对经典的强化学习算法进行修改，分别实现了基于 DQN 的推荐算法、基于 DDPG 的推荐算法以及基于 Reinforce 的推荐算法。分析了以 Movielens-100k 数据集

进行训练时，强化学习推荐算法出现误差的来源。

(3) 针对使用离线数据集训练强化学习推荐算法容易出现训练误差的问题，提出一种将模仿学习与强化学习相结合的推荐算法，通过在训练过程中将状态与动作进行约束来限制强化学习产生误差。

(4) 针对模仿学习与强化学习在同一个网络训练时，存在训练效果不稳定不可控的问题，本文提出一种解耦网络的改进训练方法，通过使用两个网络分别训练强化学习和模仿学习，并使用一个参数来调节模仿学习对于强化学习动作选择范围的限定程度，以此来控制训练效果的偏向。

(5) 通过在全新数据集上再次实验来测试本文提出的推荐系统的效果，最终验证了本文所提出的推荐系统以及改进训练方法的有效性。

## 1.4 本文结构安排

本文的结构安排如下：

第一章，对推荐系统的发展过程进行了介绍，并分析了当前推荐系统存在的问题。

第二章，对推荐系统进行概述，详细介绍几种传统与现代的推荐方法。

第三章，阐述部分强化学习理论，分别介绍了三种不同类型的强化学习方式。

第四章，提出基于强化学习的电影推荐系统框架的设计方法，并搭建模型，获取电影嵌入向量。分析实验出现错误的原因，并提出了一种能够有效解决推荐系统出现误差的方法。同时提出一种解耦网络的改进训练方法来对强化学习推荐算法进行训练。

第五章，介绍了实验环境以及数据集处理的具体方法，对第四章提出的模型和各种改进方法在 MovieLens 数据集进行实验并验证效果，对实验效果进行分析。最后在一个新的数据集上进行系统测试。

## 第二章 推荐方法概述

本章对推荐系统发展过程中的主要方法进行了介绍，依次详细叙述了传统的协同过滤方法和近几年流行的深度学习方法。

### 2.1 协同过滤方法

2005 年以前，推荐系统的主要研究方向集中在如何利用用户-物品 (U-I) 矩阵方面，该矩阵编码了用户对集合中物品的个人偏好，并为协同过滤 (CF) 技术提供了基础，而协同过滤技术是推荐系统领域的一种重要方法<sup>[35][36]</sup>。

#### 2.1.1 基于内存的协同过滤技术

基于内存的协同过滤算法使用用户-物品数据库的整个或部分样本来生成一个推荐预测。每个用户都属于某一群有着相似兴趣的人，都存在部分人与他爱好相似。通过识别新用户（或活跃用户）的相似邻域人群，可以为他或她预测其对新物品的偏好。邻域协同过滤算法作为一种最普通的协同过滤类方法，它利用所有用户对物品的评分记录来计算用户（或物品） $i$  和  $j$  之间的距离、相关性或权重  $w_{i,j}$ ，从而为活动用户产生预测。当推荐任务是生成一个 Top-N 推荐列表时，我们需要在计算相似性后找到  $k$  个最相似的用户或物品，然后收集这些物品，得到 Top-N 个最相似的物品作为推荐。

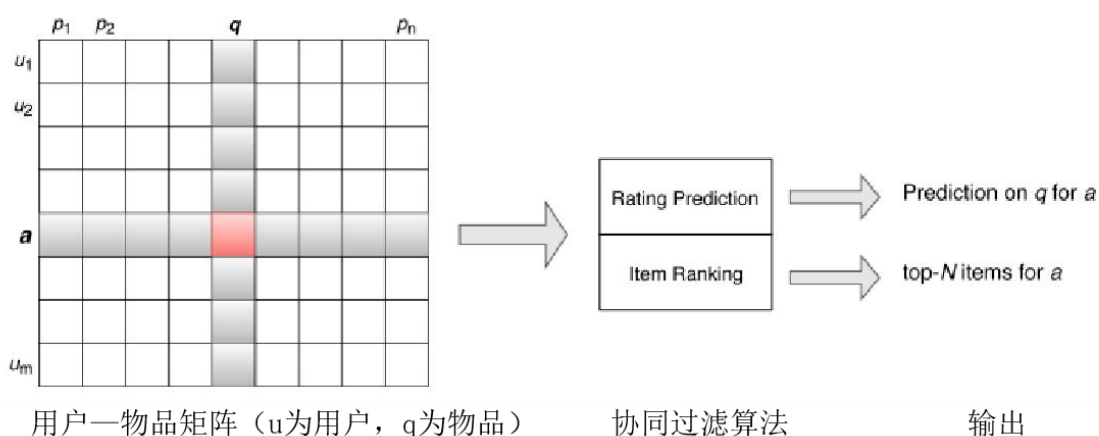


图 2-1 协同过滤过程概述

Fig.2-1 An overview of the CF process

基于内存的协同过滤算法需要计算物品或用户之间的相似性。例如如果想要确定两

个物品是否相似，首先需要找出两个物品的共同用户，然后通过一些计算方法，将这些用户对物品的评分进行计算，从而确定两物的相似程度。下面介绍几种不同的计算相似度的方法。

### （1）基于相关性的相似性

在这种情况下，两个用户  $u$  和  $v$  之间的相似性  $w_{u,v}$ ，或两个物品  $i$  和  $j$  之间的  $w_{i,j}$ ，是通过计算皮尔逊相关性或其相关性来测量的。

皮尔逊相关性衡量两个变量之间线性关系的程度<sup>[38]</sup>。

对于基于用户的算法，用户  $u$  和  $v$  之间的皮尔逊相关性见公式（2.1）：

$$w_{u,v} = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}} \quad (2.1)$$

公式中的  $i \in I$  表示两个用户都打过的物品。 $\bar{r}_u$  是用户  $u$  所有评分的平均值。

对于基于物品的算法，用户  $u \in U$  包含表示对项目  $i$  和  $j$  都进行评分的用户，皮尔逊相关性见公式（2.2）：

$$w_{i,j} = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_j)^2}} \quad (2.2)$$

公式中的  $r_{u,i}$  是用户  $u$  对物品  $i$  的打分。 $\bar{r}_i$  是物品  $i$  的所有评分的平均值。

基于物品和基于用户的皮尔逊相关性的一些变体可以在<sup>[39]</sup>中找到。基于皮尔逊相关的协同过滤算法是一种具有代表性的协同过滤算法，在协同过滤研究界得到了广泛的应用。

### （2）基于向量余弦的相似性

在协同过滤中，可以使用用户或物品的评分向量形成的余弦角来衡量他们的相似性。

如果  $R$  是用户-物品矩阵，那么可以通过计算矩阵  $R$  中对应列的向量的余弦值来表示两个物品  $i$  和  $j$  的相似性。物品  $i$  和  $j$  之间的余弦相似度见公式（2.3），“ $\cdot$ ”表示两个向量的点积。

$$w_{i,j} = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\| * \|\vec{j}\|} \quad (2.3)$$

如果要获得全部的相似性,需要计算一个  $n \times n$  的相似矩阵。例如,如果向量  $A=\{x1, y1\}$ , 向量  $B=\{x2, y2\}$ , 则 A, B 之间的向量余弦相似度见公式 (2.4) :

$$w_{A,B} = \cos(\vec{A}, \vec{B}) = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| * \|\vec{B}\|} = \frac{x_1x_2 + y_1y_2}{\sqrt{x_1^2 + y_1^2} \sqrt{x_2^2 + y_2^2}} \quad (2.4)$$

有时候不同的用户打分的习惯不同,这是向量余弦相似度无法考虑的,这时候需要从每个评分对中减去相应用户的平均分。

### 2.1.2 基于模型的协同过滤技术

通过对模型的训练可以让系统获得较强的识别能力,然后可以使用模型完成许多复杂的任务,例如对协同过滤任务进行预测。基于模型的协同过滤算法可以解决一些其他模型存在的问题,比如可以应对普通的协同过滤算法在数据稀疏时无法进行推荐的情况。

#### (1) 基于回归的协同过滤算法

回归方法对评分使用回归模型近似来进行预测。定义  $X=(X1, X2, \dots, Xn)$  是一个随机变量,表示用户对不同物品的喜好。线性回归模型可以表示为:

$$Y = \Lambda X + N \quad (2.5)$$

其中  $\Lambda$  是一个  $n \times k$  的矩阵。 $N=(N1, \dots, Nn)$  是一个代表用户选择噪声的随机变量, $Y$  是一个  $n \times m$  矩阵,  $Y_{ij}$  是用户  $i$  在物品  $j$  上的评分,  $X$  是一个  $k \times m$  矩阵,每个列作为一个用户的随机变量  $X$  (用户在  $k$  维评分空间中的评分) 的值的估计。通常,矩阵  $Y$  是非常稀疏的。

#### (2) 基于 MDP 的协同过滤算法

Shani 等人认为推荐过程是一个顺序优化问题,而不应该将它视为预测问题,并在推荐系统中使用了马尔可夫决策过程(MDPs)模型<sup>[45]</sup>。

马尔可夫决策过程通常用于有关智能体通过动作影响其周围环境的问题中。马尔可夫决策过程可以表示成  $(S, A, R, P)$ , 其中  $S$  是一系列状态,  $A$  是一系列动作,  $R$  是对于每个状态/动作对的奖励函数,  $P$  是环境的转移概率<sup>[44]</sup>。

对 MDP 的最优解是使其总奖励值最大化。从初始策略开始, 计算以此策略得出的奖励值, 并在每一步用奖励值的多少来更新策略, 迭代最终收敛到最优策略。

### (3) 隐语义协同过滤模型

隐语义协同过滤技术依赖统计建模技术, 它在模型中引入潜在的分类变量, 以发现用户群体和物品分类。从概念上讲, 它使用重叠的用户群体分解用户偏好。与基于内存的方法相比, 该技术的主要优点是其更高的准确性和可扩展性, 但同时也缺乏可解释性。该方法与日后兴起的神经网络方法有异曲同工之处。

#### 2.1.3 混合协同过滤技术

混合协同过滤系统将协同过滤与其他推荐技术(通常与基于内容的推荐)相结合, 以做出预测或建议<sup>[46]</sup>。

基于内容的推荐系统针对的是用户的一些私人信息、行为记录或是一些图片上的场景等内容, 它可以从这些内容中找到规律性, 许多元素有助于展现文本内容的重要性, 像观察到的单词或页面的浏览特征(例如术语频率和逆文档频率), 以及用户在过去中喜欢的物品之间的相似性, 然后, 基于内容的推荐系统使用启发式方法或分类算法来为用户进行推荐<sup>[3]</sup>。基于内容的推荐技术存在冷启动问题, 因为它们必须有足够的信息来构建一个可靠的分类器, 此外, 它们还受到与推荐对象显式关联的特征的限制(有时这些特征很难提取), 而协同过滤可以在没有任何描述性数据的情况下提出建议。基于内容的技术存在过度专业化的问题, 也就是说, 它们只能推荐那些根据用户的个人资料或他的评分历史中得分很高的物品, 为了避免推荐系统受到限制并提高推荐性能, 我们将协同过滤技术与基于内容的技术结合来组成新的推荐方式<sup>[46]</sup>。

相比单一的基于内容的推荐技术和协同过滤技术, 将两者进行混合后得到的混合推荐系统不仅提高了预测性能, 还克服了冷启动问题, 同时解决了协同过滤技术存在的稀疏性问题。

## 2.2 深度学习方法

深度学习是机器学习的一个子领域, 以层级学习表示为基础, 通常是使用人工神经网络进行训练。近年来, 由于计算机计算能力的增强和大数据存储设施的增加, 人工神经网络开始重新火热起来, 各种深度神经网络模型不断问世, 推动了深度学习的成为计算机科学的一个新兴领域<sup>[53]</sup>。目前, 在人工智能领域里, 大部分最先进技术都与

神经网络有很大关联。深度学习技术强大的能力也使得推荐系统研究人员在推荐任务中使用深度学习架构<sup>[53]</sup>。

深度学习技术广泛的使用场景在过去这几年来不断地被发掘出来。而深度学习近些年来重新火热主要原因有：

- (1) 大数据：海量训练数据的存在使得深度学习模型能够更好的进行表示。
- (2) 算力：图形处理单元(GPU)满足深度学习模型中复杂计算所需的处理能力。

深度学习技术与推荐系统的融合的是一种强强联手。深度学习强大的分析能力能够将数据隐藏的信息通通发掘出来。同时随着全球的数据存储系统不断增加和芯片计算能力的不断进步，深度学习技术展示出的强大效果也不断显现出来。因此利用深度学习解决来推荐系统的数据稀少性问题是一个很不错的选择，并且还能够提高系统的可扩展性。除此之外，深度模型还可以起到数据降维的效果，从隐含层面分析特征并使用这些特征获取推荐结果。本节简要介绍了一些基础的深度学习推荐模型。

### 2.2.1 AutoRec

AutoRec 是一种最简单的基于深度学习技术提出的推荐方法，他本身是一个自编码器，利用自编码器对用户物品矩阵进行自编码，得到自编码器的输出后根据输出对用户推荐<sup>[53]</sup>。自编码器包含编码器和解码器两个部件，数据输入后通过编码器进行编码降维，然后通过解码器恢复原本的数据维度，以这种方式来对原数据进行重新生成。在推荐系统中，可以利用它对包含缺失的用户物品矩阵进行补全，通过重新生成数据的方式对原本矩阵中缺失的值进行预测。同时自编码器的隐含层其实是一种对数据特征的深度挖掘，可以获得数据更加深层次的表示来应对推荐过程中出现的数据量不足的问题<sup>[48]</sup>。

AutoRec 利用了自编码器来解决构建重建函数的需求，从模型的结构图 2-2 中可以看出，网络的输入层是物品的评分向量  $r$ ，输出层是一个多分类层，图中中间的神经元代表模型的  $k$  维单隐层，其中  $k$  远小于  $m$ ，图中的  $V$  和  $W$  分别代表输入层到隐层，以及隐层到输出层的参数矩阵。该结构的具体形式如公式 (2.6) 所示。

$$h(r; \theta) = f(W \bullet g(Vr + \mu) + b) \quad (2.6)$$

其中， $f$  和  $g$  分别为输入层和隐层的激活函数。



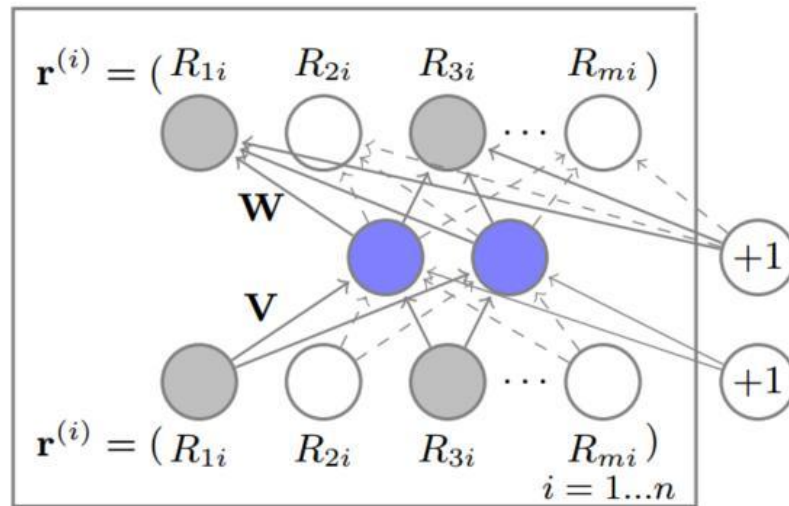


图 2-2 AutoRec 网络结构

Fig.2-2 Autorec network structure

基于 AutoRec 模型的推荐过程并不复杂，当输入物品  $i$  的评分项量  $r^{(i)}$  时，模型的输出向量  $h(r^{(i)}; \theta)$  就是所有用户对物品  $i$  的评分预测，其中的第  $u$  维就是用户  $u$  对物品  $i$  的预测  $\hat{R}_{ui}$ 。

### 2.2.2 递归神经网络

在推荐系统领域，还有一种叫做递归结构的神经网络使用最为广泛，它可以针对序列型的数据进行深度挖掘来获取其他网络无法获取的数据序列间的连续性信息，比如用户的网购过程就可以看成是一种序列型的数据，以前浏览历史中包含着最近购买物品的信息，这就是在推荐系统中使用递归结构网络的好处。将递归神经网络对时序型数据强大的处理能力用在用户购物记录，浏览记录或是评分记录上，可以取得比常规神经网络更好的效果，通过充分挖掘一些在顺序上所体现出来的特征或性质来改进推荐效果<sup>[50]</sup>。

递归神经网络可以获得序列中前后不同操作之间相关性的能力使得推荐模型可以获得长效的推荐效果。利用递归神经网络，从过程中捕捉到用户兴趣的变化，并做出不同的推荐选择。在对比传统的推荐方法与深度推荐方法时，发现传统的基于邻域或因式分解的方法的推荐效果只能存在很小段时间，不能起到长效推荐的效果，而递归

神经网络的序列学习能力可以有效得抵消掉因为推荐算法短视所造成的推荐效果僵化问题。所以将递归神经网络运用于交互型的推荐系统，利用它来捕捉用户前后操作的隐含关联并以此提高推荐效果是一个不错的选择。

### 2.2.3 GraphSAGE

GraphSAGE 的全称叫做 Graph Sample and Aggregate，翻译过来叫“图采样和聚集方法”，是一种图神经网络方法，他的名称就很好地解释了它运行的过程，就是先“采样”、再“聚集”。

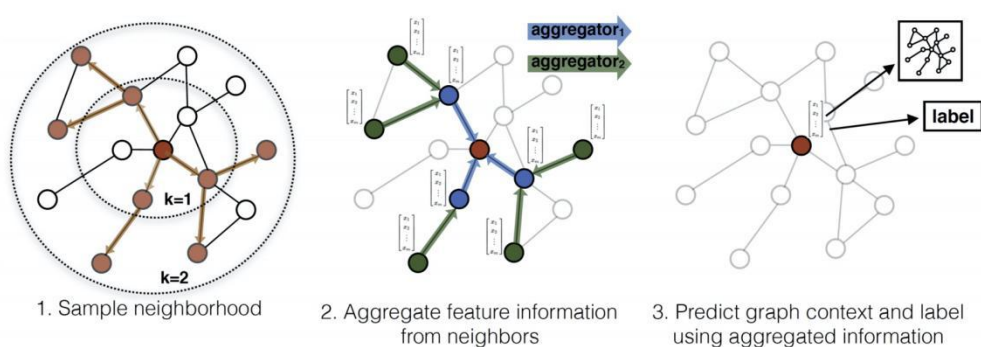


图 2-3 GraphSAGE 主要过程

Fig.2-3 Main process of GraphSAGE

GraphSAGE 的主要过程如图 2-3 所示，整个过程分为三步，首先需要在图数据上随机选择一个中心点，开始采样，得到一个二阶的子图；然后利用 GNN 将二阶子图的邻接点聚合成一阶的邻接点，再把一阶邻接点聚合成这个中心节点；有了聚合好的中心节点的 Embedding 后，就可以利用它去执行预测任务。

## 2.3 本章小结

本章首先对推荐系统发展过程中的相关方法进行了详细的介绍，为后续工作的开展提供了思路。首先，对传统的协同过滤方法进行了分类介绍与总结，依次详细介绍了基于内存和基于模型的协同过滤的概念与方法。随后介绍近几年发展火热的深度学习如何应用于推荐系统，阐述了几个较为经典的例子为后续工作提供思路。

### 第三章 强化学习概念与理论介绍

本章介绍了强化学习的基本概念和相关理论，浅要地对强化学习方法进行了分类，并详细介绍部分方法具体的细节与发展过程。

#### 3.1 强化学习概念

强化学习用于在交互过程中寻求最优策略，以此在整个过程中得到最多的奖励或者实现某个目的<sup>[52]</sup>。强化学习的灵感来源于人类的进化过程。人类在历史长河中与自然环境不断交互、学习和积攒经验，不断尝试用不同的方式来解决各种事情，在没有先验指导的情况下，通过不断探索，根据结果的好坏，来推进任务朝着可以完成的方向推进<sup>[23]</sup>。

可以将强化学习的思想总结成一种解决未知问题的通用步骤，即在整个过程中的不断探索和获得经验<sup>[23]</sup>。

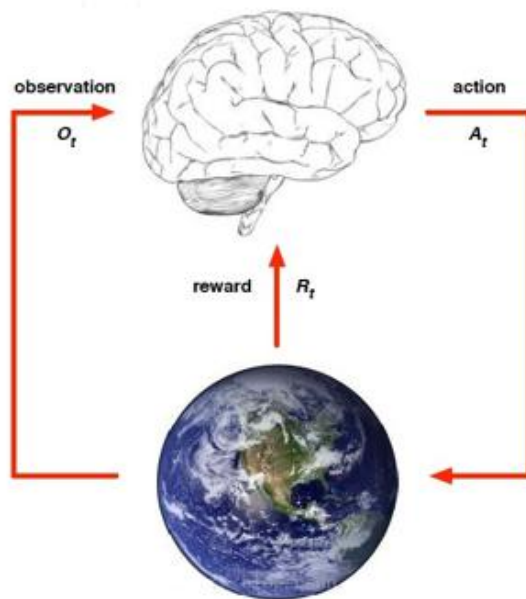


图 3-1 人类与自然环境交互的过程

Fig.3-1 The process of human interaction with the natural environment

强化学习一般应用在具有马尔可夫性的序列过程中，根据对环境了解程度的不同，分为有模型的强化学习和无模型的强化学习。解决方法分为两种思路，一条路是动态规划方法，一般用于解决有模型类的强化学习问题，另一个方向是随机采用法，通过

不断试探，从经验中获得平均值。动态规划方法可以分为策略迭代与值迭代两类，随机采样多是采用蒙特卡洛方式。更细的，从策略学习方式上可以分为同步与异步两种方式，例如同步学习的有 SARSA 算法、Policy Gradient 算法，异步学习的有 Q-learning，从学习的目标又可以分为基于值的方法和基于策略的方法两种<sup>[23]</sup>。

强化学习中有一些常用的指代名词，包括智能体、环境、状态和动作等，具体的含义解释如下：

(1) 智能体：智能体是做出交互动作的物体，它与环境相对应，例如在真实环境中人是智能体，在游戏中控制单位为智能体，在电影推荐系统中推荐系统即为智能体，它负责为人们推荐电影。

(2) 环境：环境接受智能体发出动作并对动作做出反馈，同时返回一个新的状态给智能体，在电影推荐系统中，环境的角色是人。

(3) 动作 ( $A$ )：动作是智能体所作出的行为，即智能体与环境进行交互的行为，动作的概念十分宽泛，可以是简单的离散动作，例如游戏中控制单位的前进后退，电影选择的中的电影 id，也可以是电流、电压，车的行驶速度等连续型动作，所有动作的集合称为动作集合  $A$ 。

(4) 状态 ( $S$ )：状态是对环境的描述，每一个时刻环境都有一个状态，环境通过将新状态和奖励返回给智能体来达到交互目的。在电影推荐系统中，人的观影历史即为某一时刻的状态。

(5) 奖励 ( $R$ )：奖励是环境反馈给智能体的一个数值，环境接收到智能体的动作后给智能体一个反馈，根据反馈数值的好坏，智能体调整自身策略，在电影推荐系统中可以认为人的打分是一种奖励。

(6) 策略 ( $\pi$ )：指智能体在面对不同状态时选择动作的方式。

(7) 折扣因子 ( $\gamma$ )：折扣因子作用于环境对智能体的反馈奖励上，是一个超参数，它的大小控制智能体对于当前收益与长远收益的态度，折扣因子越接近 1 表示对未来的收益越看重，折扣因子越接近 0 则表示智能体更看重当前的立即收益<sup>[23]</sup>。

(8) 轨迹：轨迹是指一连串智能体与环境交互的历史记录，通常用以下的记录方式表示：

$$[s_0, a_0, r_1, s_1, a_2, \dots, s_t, a_t] \quad (3.1)$$

(9) 转移概率(P):  $P(s_{t+1} | s_t, a_t)$  定义了环境在状态  $s_t$  选择动作  $a$  后转移到状态  $s_{t+1}$  的概率, 对于马尔可夫决策过程有  $P(s_{t+1} | s_t, a_t, \dots, s_1, a_1) = P(s_{t+1} | s_t, a_t)$ 。

强化学习问题的过程可大多具有马尔可夫性。马尔可夫性原文描述意思为在一个随机过程中, 当前的状态仅与它之前一个状态有关, 而与过去的其他状态都无关。

### 3.2 强化学习方法

强化学习包含很多种方法, 根据有无具体环境模型, 我们可以分为无模型和基于模型的方法。其中基于模型的方法需要对环境十分的了解, 而无模型方法不需要, 所以依照任务是否环境可知进行方法选择。由于推荐系统环境是不同的人, 属于环境不确定, 所以我们在无模型的方法里面进行选择。

强化学习根据策略选择方式的不同, 可以分为两大类。基于策略类的方法作为一种最直观的强化学习方法, 可以在于环境交互过程里直接学习动作的好坏, 通过给出概率值来表示每一个动作是否值得做, 以此方式来选择动作。值函数类的方法通过执行不同的动作, 不断迭代其价值, 最终以此为选择动作的依据。与策略类方法相比, 值函数类的策略选择更为确定, 只选动作价值最高的, 而策略类的方法是依据每个动作的概率来选择, 所以基于策略的方法更适合石头剪刀布这中随机策略类的问题。

#### 3.2.1 基于值的方法

智能体的最终目的是使得整个过程的累计期望收益最大化, 即:

$$\max \left( \sum_t \gamma^t r_t \right) \quad (3.2)$$

那么智能体的优化目标就可以设置为在当前状态为  $s_t$  的时候, 选择合适的动作  $a_t$  使得之后的累计收益最大化, 不同值函数的表达式见公式 (3.3) 和公式 (3.4)。

$$V_{\pi}(s) = E_{\pi} \left[ \sum_t \gamma^t r_t \mid s \right] \quad (3.3)$$

$$Q_{\pi}(s, a) = E_{\pi} \left[ \sum_t \gamma^t r_t \mid s, a \right] \quad (3.4)$$

策略  $\pi$  下状态  $s$  的价值函数记为  $V_{\pi}(s)$ ，即状态值函数（state value function），代表从状态  $s$  开始，智能体按照策略进行决策所获得的总回报。类似的，根据策略  $\pi$  在状态  $s$  下采取动作  $a$  的后决策序列的总回报记为  $Q_{\pi}(s, a)$ 。

无模型的强化学习主要有两种方法来实现更新策略：

（1）蒙特卡洛（Monte Carlo, Mc），该方法是指从交互开始模拟运行，直到最终状态结束一幕，计算这些轨迹的累计收益，即通过多次采样运行轨迹来计算累计收益，该累计收益可以由以下公式（3.5）表示，然后将采样得到的累计收益  $G_t$  带入公式（3.6）中计算更新状态值函数  $V(s_t)$ 。

$$G_t = R_t + \gamma R_{t+1} + \dots + \gamma^{T-1} R_T \quad (3.5)$$

$$V(s_t) \leftarrow V(s_t) + \alpha (G_t - V(s_t)) \quad (3.6)$$

（2）时序差分（TD），时序差分法与蒙特卡洛法不同，时序差分是指从当前状态开始模拟运行一步，然后获取这一步的短期收益和下一状态的值函数之和  $R_t + \gamma V(s_{t+1})$ （作为目标值）来更新状态值函数  $V$ ，更新公式为：

$$V(s_t) \leftarrow V(s_t) + \alpha (R_t + \gamma V(s_{t+1}) - V(s_t)) \quad (3.7)$$

以 TD 方法为基础的 Q-Learning 是一个典型的基于值的算法，该算法的主要目标是构建  $Q$  值表， $Q$  值表记录了智能体在各种不同状态下，采用不同的动作后会获得的累计回报值。不断更新  $Q$  值表的值，并且通过具有一定的随机性的策略来指导智能体选择动作，从而达到一个新的状态，不断重复这个过程直到算法训练完成。

$Q$  值表的构建过程就是强化学习算法学习的过程。刚开始， $Q$  值表的内容全部为零或者随机设定，代表刚开始的时候，智能体会随机探索选择动作。开始交互后，智能体当前所处的状态为  $s$ ，采取动作  $a$ ，到达一个新的状态  $s'$ 。算法会计算两个不同的  $Q$  值，称为  $Q$  现实和  $Q$  估计。 $Q$  估计即为  $Q(s, a)$ ， $Q$  现实是采取动作  $a$  之后可以获得的所有价值， $Q$  现实的计算通过公式（3.8）：

$$Q_t = R + \gamma * \max(Q(s', a')) \quad (3.8)$$

公式中的  $R$  是指状态到达  $s'$  时环境给智能体的反馈奖励，整个算法的执行过程中只有  $R$  对结果有着直接影响，所以奖励的设计对于决策过程至关重要。 $\gamma$  是折扣因子，代表采取动作后获得的价值在当前价值中的占比。得到  $Q$  现实后通过公式 (3.9) 计算  $Q$  目标与  $Q$  估计的差值，来更新  $Q$  值表：

$$Q(s, a) = Q(s, a) + \alpha[R + \gamma * \max(Q(s', a')) - Q(s, a)] \quad (3.9)$$

其中  $\alpha$  代表学习率，经过大量的交互计算后，就可以构建出完整的  $Q$  值表。从而使策略收敛，智能获得最优策略。

### 3.2.2 深度决策网络 DQN

由于 Q-Learning 是一种表格记录形式的强化学习算法，它的核心是需要构建  $Q$  值表，然而在面对更为复杂的问题时， $Q$  值表的数值数量将会因为状态动作的数量变得巨大而一同变得巨大，尤其是大数据时代的来临，许多的问题分解后都面临着大规模的状态动作，有些连续型问题甚至都无法通过常规的  $Q$  值表构建。与此同时，在硬件方面，对于计算机的内存要求也会变得十分苛刻，大范围的状态动作导致搜索效率也十分低下。提出解决此问题的方法就是使用深度决策网络。

深度决策网络 (Deep Q Network) 是深度强化学习时代提出的强化学习算法。DQN 利用神经网络来应对大规模的状态，通过神经网络来记忆所有的状态，通过输入状态给神经网络直接得出所有动作的  $Q$  值，免去了  $Q$  值表的构建，适合更复杂问题得求解。

利用深度学习网络来记录状态的方法在刚提出时遇到了一些问题：

- (1) 神经网络是一种监督学习的算法，它需要大量的训练样本来支撑它的训练，而强化学习的过程刚开始只有比较稀疏的交互数据。
- (2) 强化学习的相邻状态之间都有着一些关联，而神经网络的训练样本之间是相互独立的。
- (3) 训练过程不稳定，数值波动大，难以收敛。

DQN 为了解决上述问题提出了两个方法：

(1) 经验回放，为了解决神经网络训练样本稀疏和独立性的问题，DQN 提出经验池的概念，将强化学习交互的过程记录蓄积在经验池内，然后随机提取一批来训练网络。

(2) 目标网络，为了解决神经网络训练不稳定，DQN 提出了目标网络的概念。目

标网络是指一个与估计网络结构相同但是参数不同的网络，具体区别体现在，估计网络的参数是实时更新的，而目标网络的参数则是延后通过估计网络的参数更新的。通过公式（3.10）更新估计网络。由于目标网络参数的延后性，神经网络的训练更稳定。

$$Q_{eval}(s, a, \theta) = (1 - \alpha)Q_{eval}(s, a, \theta) + \alpha[R + \gamma * \max(Q_{tar}(s', a', \theta'))] \quad (3.10)$$

### 3.2.3 基于策略的方法

基于值的方法在应用于解决现实生活时，存在着两个问题：

（1）基于值的方法在应对动作时，需要得出所有动作的动作价值才能确定策略，计算收敛很繁琐。

（2）最终学习得到的是一个确定性策略，不适合解决随机性策略的问题。

基于策略的方法则可以直接对策略进行建模，对于一个状态  $s$ ，会输出该状态下选取每一个动作的概率，从而根据概率分布来选取动作。最终目标是最大化整个过程的期望收益。

$$\max E_{\tau \sim \pi} [R(\tau)], \text{ where } R(\tau) = \sum_{t=0}^{|\tau|} r(s_t, a_t) \quad (3.11)$$

Reinforce 是一种策略类的强化学习算法，算法需要智能体在环境中产生至少一整幕的交互过程，然后分别计算这些交互过程中的折扣奖励，最后利用这批数据来更新一次策略。

算法流程如下：

算法 1：

- 1：输入：可微分的策略参数  $\theta$
- 2：超参数：学习率  $\alpha$ ，奖励折扣  $\gamma$ ，更新一次策略所需要的 episode 数  $M$
- 3：动作策略：策略  $\pi(a|s, \theta)$
- 4：评估策略：策略  $\pi(a|s, \theta)$
- 5：初始化策略函数  $\theta$
- 6：循环执行：
  1. 以动作策略  $\pi$  与环境进行交互，并循环执行  $M$  个 episodes，获取  $M$  个长度



为  $T$  的轨迹数据，轨迹数据表示为：

$s_0^m, a_0^m, r_0^m, s_1^m, a_1^m, r_1^m, \dots, s_{T-1}^m, a_{T-1}^m, r_{T-1}^m, s_T^m$  其中，其中  $m \in [1, M]$

2. 计算折扣奖励：  $R_t^m = \gamma^0 r_t^m + \gamma^1 r_{t+1}^m + \gamma^2 r_{t+2}^m + \dots + \gamma^{T-1} r_{T-1}^m$ ，其中  $t \in [0, T-1]$

3. 规整化奖励：  $R_t'^m = (R_t^m - \text{mean}(R_0^m, R_1^m, R_2^m, \dots, R_{T-1}^m)) / \text{std}(R_0^m, R_1^m, R_2^m, \dots, R_{T-1}^m)$

4. 更新参数：  $\theta \leftarrow \theta + \alpha \sum_{m=1}^M \sum_{t=0}^{T-1} R_t'^m \nabla \ln(\pi(a_t^m | s_t^m, \theta))$

5. 如果满足终止条件则中断退出。

梯度更新中的  $\nabla \ln(\pi(a_t^m | s_t^m, \theta))$  是方向向量，表示从对整个轨迹的计算发现，下一次在这个方向上进行参数更新，能增大或者降低这条轨迹的出现概率。梯度里的另一块  $R_t'^m$  是一个数值，代表这一次轨迹更新力度有多大， $R_t'^m$  越大，更新力度越大，这条轨迹中的动作就越容易再次出现。由此可以看出策略类方法的目的就是不断增大高收益动作出现的次数。

### 3.2.4 Actor-Critic 算法

强化学习中还有一种算法类型，就是结合了 Q-Learning 和 Policy Gradient 算法思想而产生的 Actor-Critic 算法。

Actor-Critic 算法中的 Actor 是以动作概率为基础的策略网络，Critic 网络是 Q 值估计网络。Critic 网络充当的是一个评论家的角色，它的目的是从现实的反馈中学习如何打分，构建 Q 值。Actor 根据反馈来调整网络的输出，做出不同的动作，来让动作策略网络的动作可以获得 Critic 更高的打分。

不过 Actor-Critic 架构虽然结合了两中强化学习方法，但是其两个神经网络的参数是在连续的状态中更新的，参数存在着相关性，并且在估计值函数时，TD 目标是和估计值同时变化的，容易导致难以收敛的问题。

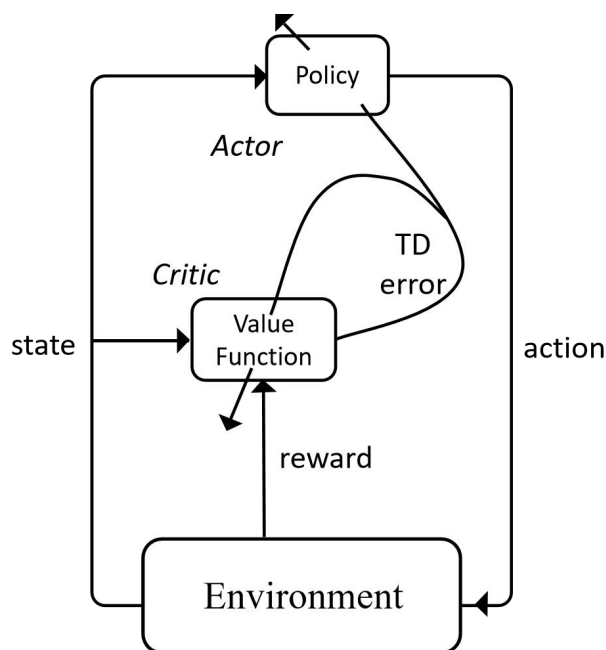


图 3-2 Actor-Critic 结构图

Fig.3-2 The Actor - Critic structure

### 3.3 本章小结

本章首先介绍了有关强化学习的含义以及由来,对于强化学习的一些基本定义和概念进行了介绍。通过对强化学习方法进行分类,分别详细介绍了值方法、策略方法、以及两种方法相结合的一种框架算法。对于值方法,首先介绍了值方法的目标以及优化方法,随后对 Q-learning 算法的工作原理与更新方法进行介绍,同时指出了不足之处。随后介绍了 DQN 算法来解决 Q-learning 算法所具有的一些缺陷。然后介绍了一种策略方法 Policy Gradient,介绍了算法的流程并指出其与值方法的本质区别。最后介绍了一种名为 Actor-Critic 方法以及它的工作的原理。

## 第四章 特征处理与模型设计改进

本章主要介绍了论文所使用的数据来源和电影嵌入向量预处理做法,以及几种基于强化学习的推荐算法的流程,同时对算法实验的结果进行分析并提出改进方法。

### 4.1 系统框架设计

本文的推荐系统主要分为两个部分,如图 4-1 所示,包括样本特征提取部分以及强化学习推荐算法部分。其中特征提取部分是为了得到所有电影的嵌入向量,嵌入向量可以理解为是电影在电影向量空间的一种表示,不同嵌入向量在向量空间的距离越近,这两部电影就越相似,距离越远,则它们就越不同,而强化学习算法部分则是按照用户在数据集的交互记录发生的时间先后将处理完的数据集输入训练,需要预先对数据集进行处理,形成一种模拟与用户一步一步交互的过程,然后按批次输入强化学习推荐算法进行模型训练,得出推荐结果。

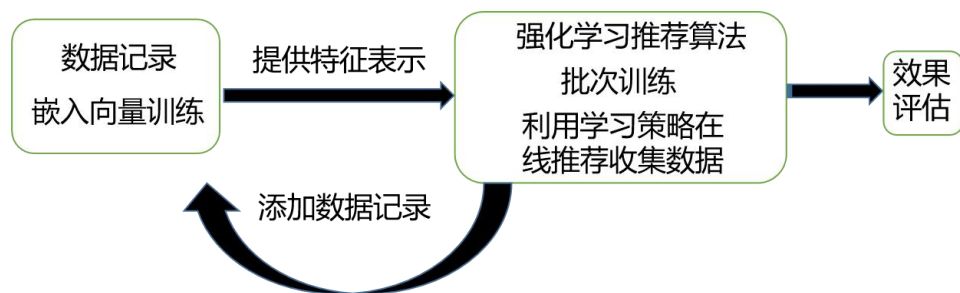


图 4-1 系统框架组成图

Fig.4-1 System framework

### 4.2 特征嵌入表示

特征嵌入又称 Embedding,其含义为‘向量化’或者‘向量映射’,指用一个低维稠密的向量来表示一个对象。在整个深度学习框架下,特别是广告、推荐、搜索等为核心的互联网领域,嵌入技术的应用十分广泛,甚至可以称其是基础核心操作。本文使用 Word2vec 模型来对每部电影做一个简单嵌入操作。

#### 4.2.1 数据集选取

为了测试本文基于强化学习的推荐系统的有效性,本文选取 Movielens-100k 数据集用于后续实验。本数据集包含三个表分别为用户表、电影表、以及交互评分表,实验

主要使用其中的交互评分表，详细数据如表 4-1 所示。

表 4-1 Movielens-100k 数据集信息

Table 4-1 Movilens-100k dataset information

Movielens-100k	
用户数	943
电影数	1682
交互记录条数	100000
用户最短交互长度	20
用户最长交互长度	737
评分范围	1-5

### 4.2.2 Word2vec 训练

Word2vec 算法是一种词向量嵌入算法，其本质是通过对语料库的句子进行训练，通过最大化每个词与相邻词同时出现的概率，得出相近单词在词向量空间的相似表示。同理，对于电影推荐系统里有交互序列的电影样本也可以使用 Word2vec 模型来训练电影间的相似性。本文采用 Skip-gram 模型结构来对电影交互序列进行训练，Skip-gram 是通过一个词来预测词周边的其他词，其模型如图 4-2 所示。

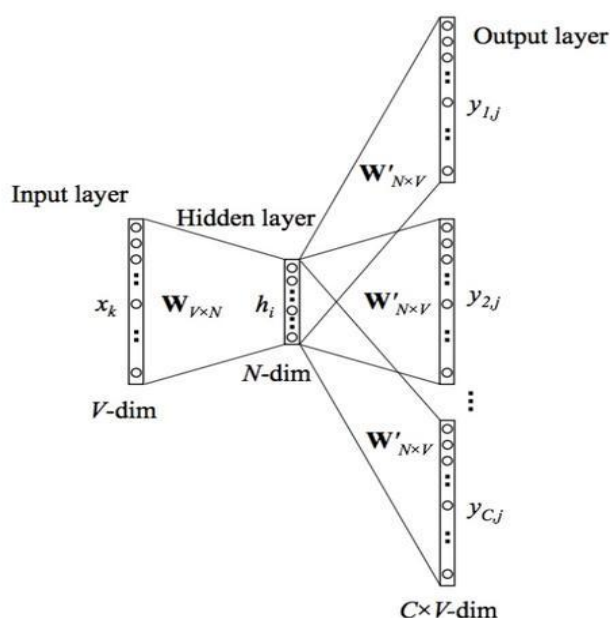
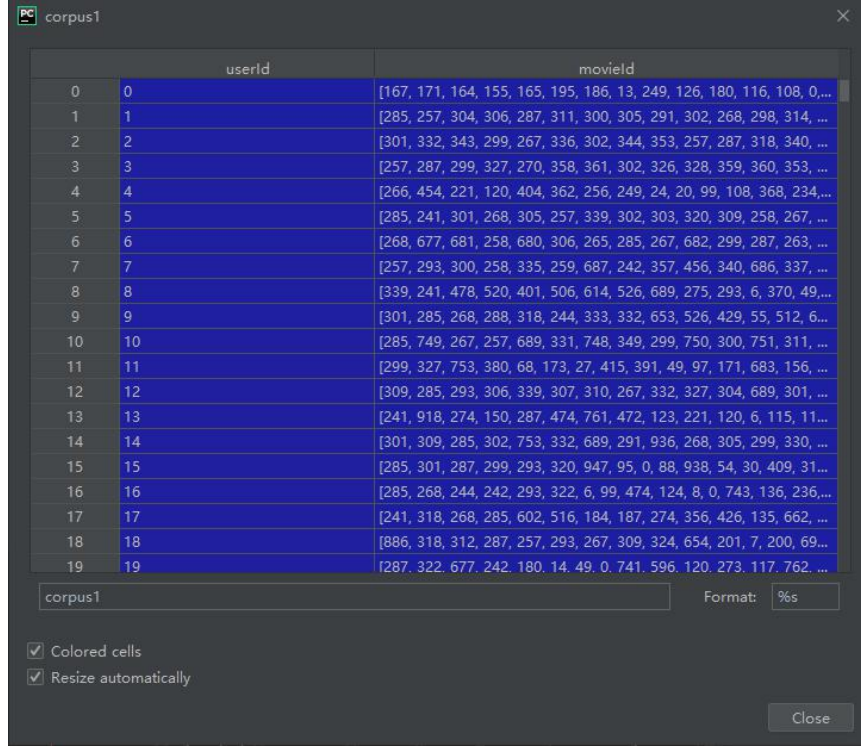


图 4-2 Skip-gram 模型

Fig.4-2 Skip-gram model

进行电影嵌入学习，首先需要做的是整理交互评分表里面的数据，将每个用户的交

互记录按照时间顺序聚合在一起，然后需要剔除评分低于 3 分的电影交互记录，因为评分较低的电影一定是在某些方面与用户打分较高的电影不同的。然后就可以得到如图 4-3 所示的交互序列记录。



	userid	movielid
0	0	[167, 171, 164, 155, 165, 195, 186, 13, 249, 126, 180, 116, 108, 0, ...]
1	1	[285, 257, 304, 306, 287, 311, 300, 305, 291, 302, 268, 298, 314, ...]
2	2	[301, 332, 343, 299, 267, 336, 302, 344, 353, 257, 287, 318, 340, ...]
3	3	[257, 287, 299, 327, 270, 358, 361, 302, 326, 328, 359, 360, 353, ...]
4	4	[266, 454, 221, 120, 404, 362, 256, 249, 24, 20, 99, 108, 368, 234, ...]
5	5	[285, 241, 301, 268, 305, 257, 339, 302, 303, 320, 309, 258, 267, ...]
6	6	[268, 677, 681, 258, 680, 306, 265, 285, 267, 682, 299, 287, 263, ...]
7	7	[257, 293, 300, 258, 335, 259, 687, 242, 357, 456, 340, 686, 337, ...]
8	8	[339, 241, 478, 520, 401, 506, 614, 526, 689, 275, 293, 6, 370, 49, ...]
9	9	[301, 285, 268, 288, 318, 244, 333, 332, 653, 526, 429, 55, 512, 6, ...]
10	10	[285, 749, 267, 257, 689, 331, 748, 349, 299, 750, 300, 751, 311, ...]
11	11	[299, 327, 753, 380, 68, 173, 27, 415, 391, 49, 97, 171, 683, 156, ...]
12	12	[309, 285, 293, 306, 339, 307, 310, 267, 332, 327, 304, 689, 301, ...]
13	13	[241, 918, 274, 150, 287, 474, 761, 472, 123, 221, 120, 6, 115, 11, ...]
14	14	[301, 309, 285, 302, 753, 332, 689, 291, 936, 268, 305, 299, 330, ...]
15	15	[285, 301, 287, 299, 293, 320, 947, 95, 0, 88, 938, 54, 30, 409, 31, ...]
16	16	[285, 268, 244, 242, 293, 322, 6, 99, 474, 124, 8, 0, 743, 136, 236, ...]
17	17	[241, 318, 268, 285, 602, 516, 184, 187, 274, 356, 426, 135, 662, ...]
18	18	[886, 318, 312, 287, 257, 293, 267, 309, 324, 654, 201, 7, 200, 69, ...]
19	19	[287, 322, 677, 242, 180, 14, 49, 0, 741, 596, 120, 273, 117, 762, ...]

图 4-3 用户交互序列记录

Fig.4-3 User interaction sequence record

然后对这部分数据分别进行滑动窗口选目标电影获取训练样本，选取窗口为左右各 20 个，不足的用一个额外的空字符来填充。获得所有的训练样本后，可以将他们分批输入如图 4-2 所示的模型进行训练。设置每个词的嵌入维度为 15，训练的目标是最大化每个词对共现的概率，见公式（3.11）：

$$\frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t) \quad (3.11)$$

其中  $w_t$  为目标电影， $w_{t+j}$  为相邻电影， $p(w_o | w_l) = \frac{\exp(v_{wo}^T v_{wi})}{\sum_{w=1}^W \exp(v_w^T v_{wi})}$ ， $v_{wi}$  与  $v_{wo}$  分别为权重矩阵  $W_{V \times N}$  与  $W'_{N \times V}$  在对应电影索引所在位置的那一条数据。

最后得到的模型的权重矩阵  $W_{V \times N}$  即为所有电影的嵌入向量的矩阵，该矩阵的每一行对应为一部电影的嵌入向量。

### 4.3 基于强化学习的推荐算法

本文研究的内容是基于强化学习的推荐系统,然而将强化学习算法运用于推荐系统存在一个基本的问题,就是无法实时得获取推荐系统与用户大量的交互信息,而强化学习的训练又依赖于这些信息。因此本文的观点是:如果想要将强化学习融合于推荐系统,那么训练所需要的数据一定是之前收集的历史数据交互记录,所以本文的强化学习推荐算法都是基于使用历史交互数据集进行训练来设计的。

本节主要介绍在使用数据集对强化学习推荐算法进行训练这一基础设定下,如何具体实现基于各类常规强化学习的推荐算法。算法具体实验效果见第 5.3.1 小节。

#### 4.3.1 基于 DQN 的推荐算法

基于 DQN 的推荐算法与常规 DQN 算法的区别就在于如何获取训练数据。常规 DQN 是一种异步更新的强化学习算法,它通过探索性策略来收集数据存储在经验池中,通过贪婪策略来学习经验池数据,而在推荐系统中,因为只能使用历史交互数据来训练模型,所以 DQN 算法中的经验池被替代为数据集中抽取出的强化学习四元组,利用四元组与强化学习推荐算法进行交互更新。其网络结构如图 4-4 所示:

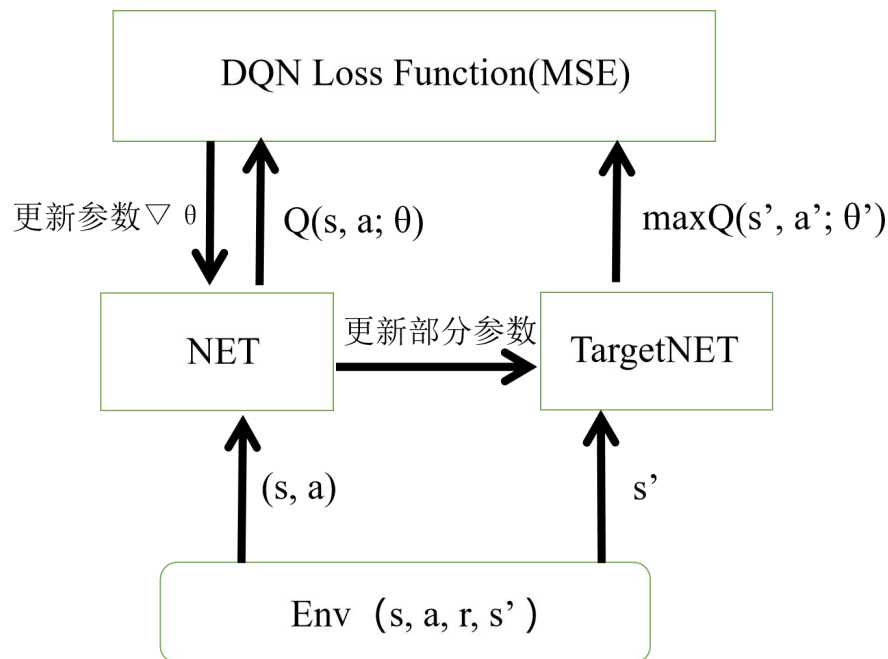


图 4-4 DQN 推荐算法网络结构

Fig.4-4 Network structure of DQN recommendation algorithm

算法流程如下：

算法 2：

- 1: 输入：数据集  $B$ ，迭代次数  $T$ ，目标网络更新率  $\tau$ ，batchsize  $N$
- 2: 初始化  $Q$  网络参数  $Q(s,a|\theta)$ ，目标网络参数  $Q(s,a|\theta')$ ， $\theta' \leftarrow \theta$
- 3: 重复  $T$  次：
- 4:   打乱数据集：
- 5:       从数据集  $B$  选取  $N$  个用户采样一批数据  $(s, a, r, s')$ ，直到取完数据集  $B$
- 6:       将  $s'$  输入目标网络，选择最大网络输出值，加上  $r$ ，计算  $Q$  目标值。  
(如果为终止状态则  $Q$  目标值为  $r$ )
- 7:       将  $s$  输入  $Q$  网络，选则动作  $a$  所在的  $Q$  值
- 8:       计算  $Q$  目标值与  $Q$  值的均方误差来更新参数  $\theta$ ，按照  
 $\theta' \leftarrow (1-\tau)\theta' + \tau * \theta$  更新  $\theta'$
- 9:   返回第 3 步
- 10: 结束

参数设置如表 4-2 所示：

表 4-2 DQN 推荐算法参数设置

Table 4-2 Parameter setting of DQN recommendation algorithm			
参数名	数值	参数名	数值
Inputsize	80	Framesize	5
Hiddensize	256	折扣率	0.99
Outputsize	1682	软更新 Tau	0.005
Embeddingsize	15	学习率	1e-5

#### 4.3.2 基于 Policy Gradient 的推荐算法

REINFORCE 是一种可以直接优化策略的强化学习算法，通过采样足够多的轨迹的方式来获取策略的优化梯度。Reinforce 算法网络结构见图 4-5。

由于我们的实验数据是从数据集中获取的，缺少了优化方法中需要的轨迹采样，即图 4-5 中的交互多幕这一步，所以只能通过收集而来的数据轨迹来尝试训练。方法是将每个用户的交互过程按时间顺序排列并认为每个用户的交互序列是一幕数据，同时将打分收集好计算出轨迹折扣奖励，模拟出交互轨迹的效果。算法流程见 3.2.3 小节算

法 1。

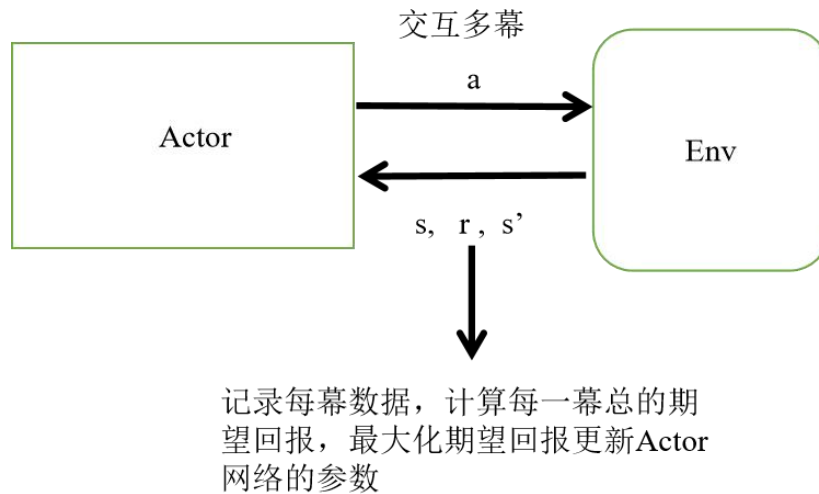


图 4-5 Reinforce 推荐算法网络结构

Fig.4-5 Network structure of Reinforce recommendation algorithm

参数设置如表 4-3 所示：

表 4-3 REINFORCE 推荐算法参数设置

Table 4-3 Parameter setting of REINFORCE

参数名	数值	参数名	数值
Inputsize	80	Framesize	5
Hiddensize	256	折扣率	0.99
Outputsize	1682	学习率	1e-5
Embeddingsize	15		

### 4.3.3 基于 DDPG 的推荐算法

DDPG 算法中的第一个 D 是指深度神经网络，在之前的 DQN 算法里使用了 Q 网络和目标网络，在 DDPG 里也使用了这些，P 是指 Policy Gradient 就是上一小节介绍的策略梯度算法，第二个 D 指的是策略梯度算法在 DDPG 里面不再是随机性策略而是确定的，这么做的优点是不再需要一条条的采样轨迹，更新快，但同时也需要像 DQN 一样进行异步采样更新。在我们的实验中，可以使用数据集代替经验池对基于 DDPG 的推荐算法进行训练。

DDPG 推荐算法网络结构如图 4-6：



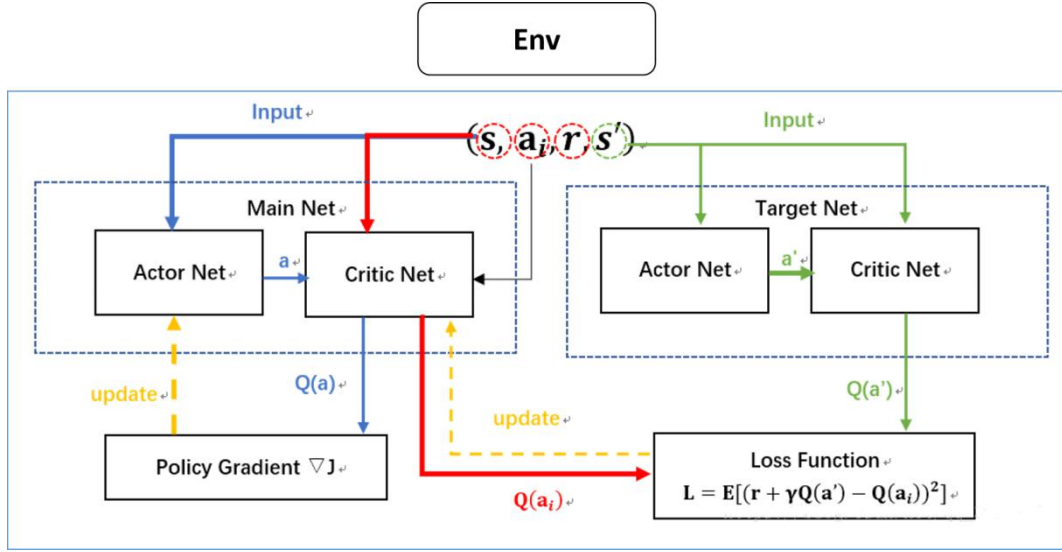


图 4-6 DDPG 推荐算法网络结构

Fig.4-6 Network structure of DDPG recommendation algorithm

算法流程如下：

算法 3：

- 1: 输入：数据集  $B$ ，迭代次数  $T$ ，目标网络更新率  $\tau$ ，batchsize  $N$
- 2: 初始化评论家网络  $Q(s, a | \theta^Q)$  和其目标网络  $Q(s, a | \theta^{Q'})$ ， $\theta^{Q'} \leftarrow \theta^Q$ ；动作网络  $\mu(s | \theta^\mu)$  和其目标网络  $\mu(s | \theta^{\mu'})$ ， $\theta^{\mu'} \leftarrow \theta^\mu$
- 3: 重复  $T$  次：
- 4: 打乱数据集：
- 5: 从数据集  $B$  选取  $N$  个用户采样一批数据  $(s, a, r, s')$ ，直到取完数据集  $B$
- 6: 将  $s'$  输入 actor 目标网络，获得每个动作的概率分布  $a'$ ，将  $s'$  与  $a'$  输入 critic 目标网络，获取  $Q$  值，再加上  $r$  计算得出  $Q$  目标值。（如果为终止状态则  $Q$  目标值为  $r$ ）
- 7: 将  $s$  与四元组中的  $a$  输入 critic 网络得到  $Q$  值，计算  $Q$  值与  $Q$  目标值的均方误差来更新  $\theta^Q$
- 8: 将  $s$  输入 actor 网络，获取每个动作当前的概率分布  $(a)$ ，将  $s$  和  $(a)$  输入 critic 网络得到输出的  $Q$  值，将  $Q$  值与动作  $(a)$  相乘后取负值作为 actor 网络的损失来更新  $\theta^\mu$ ；按照  $\theta^{Q'} \leftarrow (1 - \tau) \theta^{Q'} + \tau * \theta^Q$  和  $\theta^{\mu'} \leftarrow (1 - \tau) \theta^{\mu'} + \tau * \theta^\mu$  来更新  $\theta^{Q'}$  和  $\theta^{\mu'}$

9: 返回第 3 步

10: 结束

参数设置如表 4-4 所示:

表 4-4 DDPG 推荐算法参数设置

Table 4-4 Parameter setting of DDPG recommendation algorithm

参数名	数值	参数名	数值
ActorInputsize	80	Framesize	5
ActorHiddensize	256	折扣率	0.99
ActorOutputsize	1682	学习率	1e-5
Embeddingsize	15	CriticHiddensize	256
CriticInputsize	80+1682	CriticOutputsize	1
软更新 Tau	0.005		

#### 4.4 强化学习推荐算法的性能改进

本节针对上一节提出的推荐算法在实验过程中出现的误差来源进行具体分析,并给出相应的改进方法。

##### 4.4.1 基于 DDPG 的推荐算法的失效分析

经过有效性对比实验发现,基于 DDPG 的强化学习推荐算法训练后效果很差,多次训练对命中率提升也并不明显。需要对算法进行部分修改。

分析 DDPG 推荐算法与其他两种推荐算法的算法流程,发现在 DQN 以及 Reinforce 推荐算法中,我们都是直接采用将强化学习四元组中的动作  $a$  代入到模型对网络进行训练,例如在 DQN 算法中,我们通过选择动作  $a$  所在位置的  $Q$  值来更新  $Q$  网络,在 Reinforce 算法中则是通过将动作  $a$  的对应位置的概率取出来计算整个轨迹的回报值来更新网络,而在 DDPG 推荐算法中,在第 8 步时,是通过将 Actor 网络输出的动作分布直接用来更新动作网络,导致了此处出现了误差。

分析误差产生的原因,应该是由于在此处得出的动作分布中有许多动作在数据集中并没有出现,导致网络更新时会出现外推错误,即对于一个状态动作对,如果它没有出现在数据集中则会导致网络的更新出现随机值并且没有相关数据集数据对其进行修正,随后网络收敛到了错误的点,最终导致网络采取了错误的策略从而拉低了整个模型的推荐效果以致近乎失效。由此得出 DDPG 推荐算法在网络更新时,需要有效杜绝

数据集以外的状态动作对出现来对网络进行参数更新，具体到算法流程中的做法是在第 8 步中从 Actor 网络输出的动作概率分布中取出四元组中对应动作  $a$  所在位置的概率做为该处的动作输出来对动作网络进行参数更新。修改后的算法性能分析见后续第 5.3.2 小节。

#### 4.4.2 强化学习推荐算法误差来源分析

由上一小节基于 DDPG 推荐算法的实验误差分析可以看出，利用数据集进行训练的强化学习算法经常会在训练过程中出现因为探索数据不足而导致无法正确更新的问题，进而影响推荐算法的性能。因此想要在推荐系统中运用强化学习需要妥善解决这一类问题。

在对第 5.3.2 小节能对比实验后续结果的分析中我们发现 DQN 推荐算法的性能在多次训练后出现了严重的下滑，而相比之下，DDPG 和 Reinforce 推荐算法的性能在趋于稳定后基本不会变动。由此可以得出结论，DQN 算法在训练过程中依然存在外推错误。依照上一节的分析方法分析算法流程发现，问题出现在算法 2 的第 6 步，即这里使用了一个最大化选择动作的操作可能导致算法会选择不属于数据集中的动作进而产生错误并扩散，而在常规 DQN 算法中，最大化这一步是为了后续执行对算法目标策略的评估。

为了找出最大化操作产生误差的深层原因，本文对两种不同的强化学习训练方式的过程展开分析。经过对比分析发现使用数据集当作经验池训练 DQN 推荐算法和实时交互训练的 DQN 算法训练在过程中有一个地方是不同的，就是虽然实时交互的 DQN 算法也是一种异步更新策略的算法，但是由于是实时采集数据并训练更新目标策略，行为策略和目标策略所使用的策略分布是相同的，即是用同一个  $Q$  网络来表示策略分布，在这种情况下采样得出的奖励才会对目标策略的更新起到作用，而在使用数据集训练 DQN 推荐算法时，由于数据集是依据某些行为策略提前收集而来的，所以在利用数据集异步训练目标策略时  $Q$  网络的策略分布与行为策略的策略分布不一致（即目标策略分布与数据集的分布不一致），此时利用数据集中的奖励对 DQN 算法进行训练并不会起到正向作用，最终导致算法训练不会收敛或者收敛效果很差。

经过上述分析可以得出，利用数据集训练强化学习推荐算法相比于实时训练存在天然的缺陷有以下几点：

1. 使用数据集进行强化学习训练缺少实时交互，导致探索后得不到相关数据的修

正，使得原本对强化学习有利的探索反而会影响训练效果并产生无法消除的误差。

2. 由于是从数据集中学习最优策略，而数据集无法做到与实时交互一样提供所有情况数据的分布，导致算法学习到的最优策略分布理论上存在上限。

3. 使用数据集进行异步策略更新时，由于所学策略分布与数据采集时的分布不一致导致数据集中的奖励可能无法对更新起到作用。

正是以上几点缺陷导致了强化学习推荐算法在训练过程中出现失控，最终形成误差并使得强化学习推荐算法的效果出现了急剧降低。

#### 4.4.3 DQN 推荐算法改进方法

强化学习推荐算法由于缺少完备的实时交互环境，只能从数据集中训练出推荐策略。在这种情况下，我们想要训练出较好的推荐策略，就需要避免在训练强化学习训练过程中出现外推错误并且让数据集中的奖励能够对策略更新起到作用。现有方法中我们可以将模仿学习与强化学习相结合来同时做到这两点。

模仿学习是一种从范例中学习的监督学习方法，在强化学习推荐系统上，这些范例可以被理解为由数据集提供的强化学习四元组，其中的状态  $s$  作为特征，动作  $a$  作为标记，用这些数据构造出一组用于多分类任务的样本用于后续训练。在强化学习推荐算法中添加模仿学习后，我们可以将用户的状态与动作进行有效的强约束，从而起到防止外推误差出现的效果。同时由于使用了模仿学习训练，网络的参数会被调整为类似于在线采样数据时网络的参数，这样可以弥补目标策略分布与采样数据策略分布的偏差，使得数据集中的奖励对策略更新起作用。

针对上一小节 DQN 推荐算法多次训练后出现性能下滑的问题，提出一种将模仿学习与 DQN 相结合的强化学习推荐算法，并对其进行实验。结合的方式有两种，第一种方式是用模仿学习对  $Q$  网络进行预训练，随后利用预训练后的  $Q$  网络接着进行强化学习训练；第二种方式是让模仿学习与强化学习一同训练，做法是在算法 2 的第 8 步中，将模仿学习的交叉熵损失加上  $Q$  网络的均方差损失一同进行最小化。为了验证模型的效果，对两种方式都进行实验并与原始 DQN 推荐算法性能进行比较。具体实验过程见第 5.3.3 小节。

#### 4.4.4 解耦网络改进方法

在上一小节中提出的基于模仿学习改进的 DQN 推荐算法存在两种训练方式，其中第二种是将模仿学习的损失函数与  $Q$  网络的损失函数一同最小化来更新  $Q$  网络。用这

种方式更新网络虽然效果不错，但是由于两种损失函数是针对同一个网络进行参数更新，所以最终的推荐效果难以确定是由哪一种学习主导，并且由于模仿学习一直在跟随强化学习算法一起训练，损失更新的融合会使强化学习获取的推荐的效果不稳定，甚至会导致推荐结果的负优化。

本节针对这种由损失函数更新导致的冲突，提出了一种解耦网络的方法来解决这个问题，解耦网络的意思是设置两个网络分别针对多分类损失函数和  $Q$  值迭代的参数更新，设定  $Q$  网络用来更新强化学习的  $Q$  值迭代， $I$  网络用来进行模仿学习。在算法过程中，首先利用  $I$  网络获取下一步各个动作的概率，随后使用一个阈值  $\tau$  来控制动作的选择是偏向于强化学习还是偏向于模仿学习，如果  $\tau=0$ ，则代表  $I$  网络对  $Q$  网络动作不作任何限制， $Q$  网络可以任选动作，如果  $\tau=1$ ，则代表  $I$  网络对  $Q$  网络选择动作限制最大， $Q$  网络只能选择  $I$  网络中值最大的动作，即相当于完全的模仿学习。具体算法流程如下。

算法 4:

- 1: 输入: 数据集  $B$ , 迭代次数  $T$ , 目标网络更新率  $\tau$ , batchsize  $N$ ,  $\tau (=0.5)$
- 2: 初始化  $Q$  网络参数  $Q(s,a|\theta_Q)$ , 目标网络参数  $Q(s,a|\theta_Q')$ ,  $\theta_Q' \leftarrow \theta_Q$ ,  $I$  网络参数  $I(s,a|\theta_I)$
- 3: 重复  $T$  次:
- 4: 打乱数据集:
- 5: 从数据集  $B$  选取  $N$  个用户采样一批数据  $(s, a, r, s')$ , 直到取完数据集  $B$
- 6: 将  $s'$  输入  $I$  网络获取所有动作的概率值, 用每个动作的概率除以最高动作的概率值得到每个动作的比率, 输出其中比率大于等于  $\tau$  的动作。将  $s'$  输入  $Q$  网络, 从  $I$  网络输出的动作中选择最大的  $Q$  值, 加上  $r$ , 计算  $Q$  目标值。(如果为终止状态则  $Q$  目标值为  $r$ )
- 7: 将  $s$  输入  $Q$  网络, 选则动作  $a$  所在的  $Q$  值
- 8: 计算  $Q$  目标值与  $Q$  值的均方误差来更新参数  $\theta_Q$ , 按照  $\theta_Q' \leftarrow (1-\tau)\theta_Q' + \tau\theta_Q$  更新  $\theta_Q'$ , 计算  $I$  网络的交叉熵损失来更新参数  $\theta_I$
- 9: 返回第 3 步
- 10: 结束

超参数与 DQN 推荐算法设置相同,  $\tau$  的大小控制了推荐效果中增加模仿学习的程

度。进行推荐时也是采用  $I$  网络输出的动作来决定  $Q$  网络动作的选择范围。具体实验效果见第 5.3.4 小节。

## 4.5 本章小结

本章针对前文提出的三种强化学习算法,分别设计设计了三种基于强化学习的推荐算法用于实验环节进行验证,通过对误差的分析发现了以离线数据集来训练强化学习推荐算法这种做法本身会带有缺陷,通过对误差来源进行分析后提出将模仿学习与强化学习相结合的训练方法来对基于数据集训练强化学习推荐效果进行提升,最后介绍了一种能够将两种学习分开训练并同时起作用的解耦网络方法。

## 第五章 实验环境设置与结果分析

本章首先介绍了在利用数据集对强化学习推荐算法进行训练这一设定下,如何处理数据集中的交互序列形成强化学习需要的四元组数据,详细地讲解了利用滑窗方法对数据进行处理和拆分的具体做法,接着对基于强化学习的各种推荐算法在有效性实验、性能试验上进行对比并分析误差原因,最后对改进后的强化学习推荐算法进行实验并验证结论。

### 5.1 数据生成及评价指标

本实验所选用的数据集的组成以及相关信息已经在第四章第二节进行介绍,本节所介绍的重点是如何将 Movielens-100k 数据集中的交互序列评分表进行处理,形成本实验强化学习推荐算法训练需要的批数据集  $B$ 。

#### 5.1.1 训练数据生成

Movielens-100k 交互序列评分表读取之后如图 5-1 所示:



	userId	movieId	rating	timestamp
0	195	241	3	881250949
1	185	301	3	891717742
2	21	376	1	878887116
3	243	50	2	880606923
4	165	345	1	886397596
5	297	473	4	884182806
6	114	264	2	881171488
7	252	464	5	891628467
8	304	450	3	886324817
9	5	85	3	883603013
10	61	256	2	879372434
11	285	1013	5	879781125
12	199	221	5	876042340
13	209	39	3	891035994
14	223	28	3	888104457
15	302	784	3	879485318
16	121	386	5	879270459
17	193	273	2	879539794
18	290	1041	4	874834944
19	233	1183	2	892079237

图 5-1 用户交互评分序列

Fig.5-1 User interaction scoring sequence

可以从图 5-1 中看出每一条交互评分记录包含了一个用户、一部电影、一个打分以

及一个时间戳，此时需要将所有打分由【1-5】分平移到【-5-5】分，因为后续需要以此分数作为强化学习过程中的奖励  $r$ ，所以将打分低的电影评分改为负值以此与高分的电影产生区别。

从这些记录中可以看出每个用户都有一条属于自己的历史交互轨迹，我们可以将每个用户的交互序列按时间聚集在一起形成一个字典，字典的 key 为每个用户的 ID，value 为一个嵌套字典；嵌套字典的 key 分别为 ‘movieid’ 和 ‘rating’，value 分别是此用户的观影序列和评分序列，用户的观影序列如图 4-3 所示。

在得到每个用户的字典后，可以将用户的观影序列和评分序列进行滑窗操作，即如图 5-2 所示：

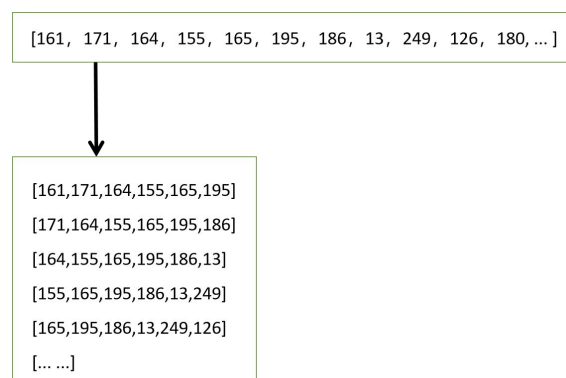


图 5-2 交互序列滑窗处理

Fig.5-2 Sliding window processing of interactive sequence

上图展示了如何对一位用户的观影序列进行滑窗操作，首先每次选定 6 个电影 ID 作为一组数据，然后依次向后滑动窗口直到最后一个电影 ID，最终将所有滑窗堆叠组成该用户的整个滑动序列。用户的打分序列做同样的处理。得到了两个堆叠的滑动序列块后，取电影 ID 序列块每一行的前 5 个电影 ID 作为索引，进入第四章得出的权重矩阵  $W_{V \times N}$  分别取出对应电影的嵌入向量（每个 15 维）拼接在一起，并加上打分序列块对应位置的 5 个打分组成一个 80 维的向量作为当前状态  $s$ ；用同样的方式可以获取后 5 个电影 ID 以及五个打分，组成另一个 80 维的向量作为后续状态  $s'$ ；取电影序列块每行最后一个电影作为动作  $a$ ，取打分序列块每行的最后一个打分作为奖励  $r$ 。至此，我们已经生成了该用户的所有四元组  $(s, a, r, s')$  数据。用同样的方式生成每一位用户的四元组数据，即可获得强化学习训练所需的数据集  $B$  了。

### 5.1.2 评价指标

将上一小节生成的每个人的数据，按照 7: 3 划分的作为训练集和验证集，取每个



人训练集的最后十条数据中的  $s$  作为测试数据，统计每次推荐的  $k$  部电影出现在验证集电影中的数目  $i_u$ 。根据推荐电影数目  $k$  的不同，依照公式（5.1）来计算推荐算法总体的命中率（hr）。

$$hr = \frac{\sum_{u \in U} \sum_{i=1}^{10} i_u}{10 * k * |U|} \quad (5.1)$$

## 5.2 实验环境

本实验采用的软硬件环境如表 5-1 所示。

表 5-1 实验环境

Table 5-1 Experimental environment

硬件环境		软件环境	
名称	型号	名称	版本
CPU	Intel i7-8750H	操作系统	Win10
内存	16GB	硬件加速	CUDA11.0
GPU	GTX 1066	深度学习框架	Pytorch1.7
硬盘	SSD128GB	语言	Python3.76

## 5.3 强化学习推荐算法对比实验

本节将对基于强化学习设计的推荐算法进行实验并分析效果，对比考虑各模型的有效性以及在不同训练批次数量、预测电影数量时的训练时长以及命中率等方面的综合表现。

### 5.3.1 有效性对比实验

首先针对第四章第三节提出的三种基础推荐算法进行有效性对比试验，目的是通过推荐结果来分析三种算法的训练流程能否有效起到推荐效果。有效性实验的对比数据如表 5-2 所示。

从有效性实验对比数据中发现，相比于 DQN 以及 Reinforce 推荐算法，DDPG 推荐算法的性能有明显的缺陷，经过对算法流程过程分析后发现出现缺陷的原因是在 DDPG 训练过程中引入了数据集以外的动作，超出了数据集的动作由于没有相关信息对其进行偏差修正，最终导致了算法收敛后的推荐效果不佳，具体分析见第 4.4.1 小节。

表 5-2 有效性实验对比数据

Table 5-2 Comparison data of validity experiment

训练轮数	N=1		
	k=5	k=10	k=20
T=18			
DQN	0.0684	0.0569	0.0515
Reinforce	0.0679	0.0683	0.0628
DDPG	0.0334	0.0300	0.0264

5.3.2 性能对比实验

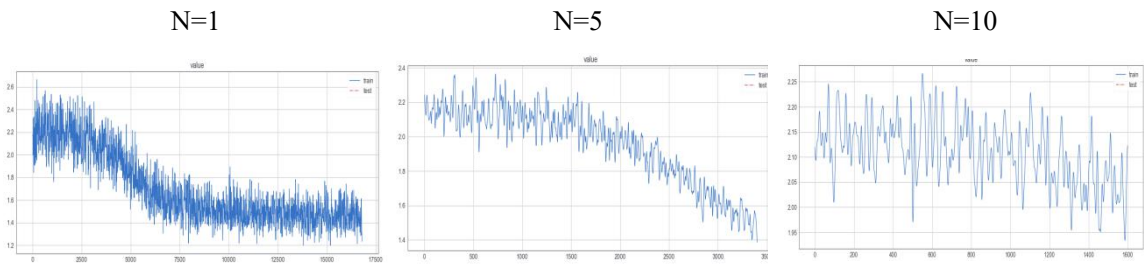
实验对经过 4.4.1 小节修改后的 DDPG 推荐算法和其他两种推荐算法采用相同的训练轮数进行训练，随后从训练时间以及命中率等方面对实验效果进行分析。训练结果的详细信息见表 5-3。

表 5-3 强化学习推荐算法性能对比

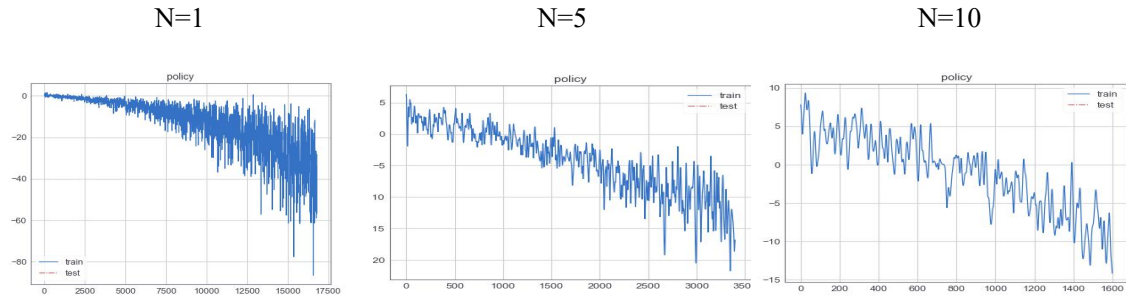
Table 5-3 Performance comparison of reinforcement learning based recommendation algorithms

训练轮数	N=1			N=5			N=10		
	k=5	k=10	k=20	k=5	k=10	k=20	k=5	k=10	k=20
T=18									
DQN	0.0684	0.0569	0.0515	0.0776	0.0696	0.0657	0.0609	0.0596	0.0528
Reinforce	0.0679	0.0683	0.0628	0.0349	0.0367	0.0366	0.0330	0.0313	0.0287
DDPG	0.0916	0.0902	0.0875	0.0882	0.0860	0.0813	0.0766	0.0764	0.0703
训练时长（秒）									
DQN	80.8			24.9			14.2		
Reinforce	419.3			90.1			57.2		
DDPG	854.1			174.6			94.2		

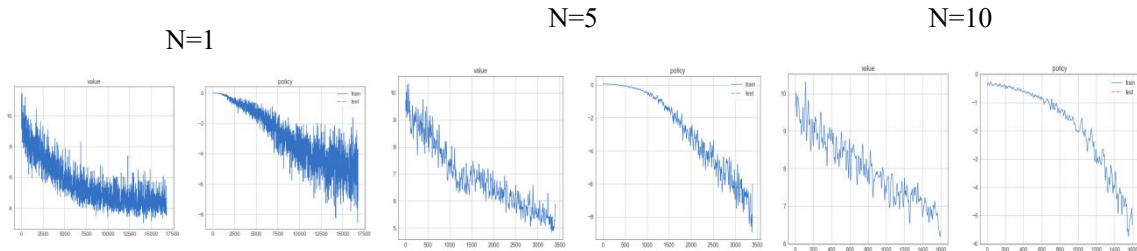
各个推荐算法在训练过程中损失的变化情况见图 5-3。



(a) DQN 推荐算法



(b) Reinforce 推荐算法



(c) DDPG 推荐算法

图 5-3 模型损失变化情况

Fig.5-3 Changes of model losses

从表 5-3 以及图 5-3 中的数据对比分析可以得出：

1. 三种基础强化学习推荐算法在离线数据集上都可以起到一定的推荐效果，并且存在区分度。
2. 从训练时长上来看，DDPG 推荐算法和 Reinforce 推荐算法的训练速度要远小于 DQN 推荐算法的训练速度。分析应该是因为 DDPG 推荐算法在训练过程中需要依次对两个网络进行更新计算，而 Reinforce 推荐算法在训练过程中分别需要计算每一幕所有的折扣奖励，导致这两种算法的训练速度不如 DQN。
3. 从损失变化图中的图像可以看出，在训练批次选 5 和 10 的情况下，各算法的训练没有进入到最终收敛的状态，需要添加一组对比实验。
4. DDPG 与 DQN 推荐算法在训练批次选 5 的情况下，命中率与训练时长可以同时取得较好的水平，而 Reinforce 推荐算法不行。分析原因是 Reinforce 算法原本的策略更新方式就是每次使用一幕的数据来对参数进行调整，同时使用多幕数据对策略梯度更新方向产生了干扰，需要多次训练才能效果较好。
5. 从训练批次选 1 的训练情况来看，Reinforce 推荐算法和 DQN 推荐算法的命中率相当都为 0.068 左右，DDPG 的命中率为 0.09 左右。随着推荐数量的增大，命中率

多数呈略微下降趋势但总体相差不大。

从以上几点分析可以看出,在训练时长上 DQN 推荐算法相对于其他两种算法具有较大的优势,在训练命中率上,目前来看 DDPG 算法目前的命中率最高。接下来增加一组实验,让所有训练方式的算法经过多次训练,再对比实验命中率,实验结果见表 5-4。

表 5-4 强化学习推荐算法命中率对比

Table 5-4 Comparison of hit rate of reinforcement learning based recommendation algorithms

训练轮数 T=72	N=1			N=5			N=10		
	k=5	k=10	k=20	k=5	k=10	k=20	k=5	k=10	k=20
DQN	0.0464	0.0481	0.0446	0.0689	0.0660	0.0657	0.0585	0.0600	0.0490
Reinforce	0.0723	0.0696	0.0631	0.0798	0.0699	0.0623	0.0418	0.0419	0.0370
DDPG	0.0934	0.0927	0.0875	0.0973	0.0946	0.0888	0.0945	0.0918	0.0905

从上表的实验数据可以看出经过多次训练后,批次为 10 的三种推荐算法中只有 DDPG 推荐算法取得了与批次为 1 和 5 基本相当的命中率,说明 DDPG 推荐算法适合使用大数量批次的训练方式,在节省时间的同时也能获得不错的推荐效果;而 Reinforce 推荐算法在批次为 10 的时候依旧没能训练收敛,所以 Reinforce 推荐算法的最佳训练批次为 5,既可以较大地提升训练速度也可以保证网络收敛。

同时实验数据显示,DQN 推荐算法在批次为 1 的情况下,多次训练后推荐效果出现了明显的下滑,分析其中原因应该是由于算法流程中存在一步选取最大值动作的操作,因此需要对推荐算法存在的误差来源进行具体分析,详细分析过程见第 4.4.2 小节。

### 5.3.3 改进强化学习推荐算法对比实验

对强化学习推荐算法的误差来源进行归纳总结后,本文提出了一种将模仿学习与强化学习相结合的推荐算法,其核心思想是通过约束状态动作对的出现种类来减少外推误差的出现并使得数据集中的奖励能够起到正向作用。实现方式有两种,一种方式为预训练 Q 网络,另一种为加入损失函数使得两种算法一起训练。改进后的 DQN 推荐算法的实验结果见表 5-5。

由表中的数据可以看出与原始的 DQN 推荐算法相比,融合了模仿学习的 DQN 推荐算法在命中率上面提升了一倍左右,性能具有明显的改进效果。因此说明模仿学习的加入成功地约束了状态动作对的出现种类,使之难以超出数据集,同时在 DQN 算法

具有最大值选取操作的情况下依然取得了高命中率，说明对于外推错误具有明显的抑制作用，并且改进后的两种训练方式都取得了比 DDPG 还要好的效果，说明数据集的奖励也对目标策略更新起到了正向作用。两种不同训练方式的结果中只有批次为 1 的命中率一直最高，说明添加模仿学习的 DQN 推荐算法适用于小批次的训练方式，同时因为 DQN 推荐算法的训练时间不长，所以选取训练批次为 1 的训练方式总体效果最佳。

表 5-5 三种 DQN 推荐算法命中率对比

Table 5-5 Comparison of hit rate of three DQN recommendation algorithms

训练轮数 T=72	N=1			N=5			N=10		
	k=5	k=10	k=20	k=5	k=10	k=20	k=5	k=10	k=20
DQN	0.0464	0.0481	0.0446	0.0689	0.0660	0.0657	0.0585	0.0600	0.0490
PreI-DQN	0.1025	0.0999	0.0899	0.0774	0.0731	0.0646	0.0787	0.0736	0.0647
I-DQN	0.1323	0.1281	0.1233	0.0869	0.0828	0.0760	0.0762	0.0745	0.0712

#### 5.3.4 解耦网络效果对比实验

本小节针对前文提出的解耦网络进行实验，测试所提出的解耦网络方法应用于强化学习推荐算法中的实际效果，并与非解耦网络在训练批次为 1 的情况下进行比较同时分析四种方法的结果。具体实验数据如下表 5-6 所示。

表 5-6 四种 DQN 推荐算法命中率对比

Table 5-6 Comparison of hit rate of four DQN recommendation algorithms

训练轮数 T=72	N=1		
	k=5	k=10	k=20
DQN	0.0464	0.0481	0.0446
PreI-DQN	0.1025	0.0999	0.0899
I-DQN	0.1323	0.1281	0.1233
解耦 $\tau=0.3$	0.1078	0.1138	0.1192
解耦 $\tau=0.9$	0.0907	0.0641	0.0456

本实验采用了两种不同的  $\tau$  值来对解耦网络方法的性能进行对比测试，从实验的结果中可以发现，解耦网络后的推荐算法与没有解耦网络的推荐算法在  $k=5$  情况下，命中率与预训练模仿学习的 PreI-DQN 算法能达到相同水平，低于同时训练的 I-DQN 算法。分析原因应该是 I-DQN 算法中模仿学习是同时训练的，所以在本文所处的离线数

据集上面测试的效果会更加好，而强化学习推荐算法需要在现实动态的环境中进行推荐，这部分的效果并不能作为 I-DQN 性能更加优秀的证明，因此解耦方法的推荐效果是能够达到基准线水平以上的。

同时从  $\tau=0.3$  的解耦方法的实验数据中可以看出，当  $\tau$  取值比较低时，算法在  $k=10$  的情况下推荐效果反而更加好，而当  $\tau=0.9$  时，算法的大数量推荐效果反而急速下滑。分析原因应该是当  $\tau=0.3$  时，由于 I 网络对于 Q 网络的控制越低，Q 网络选择的自由性越大，所以受制于模仿学习的程度就越低，从而使得 Q 网络的动作选择更为自由，最终在大数量推荐上取得了更加好的推荐效果，也是本文唯一做出大数量推荐效果提升的算法。此实验效果对比也侧面验证了解耦方法将强化学习与模仿学习分开进行参数更新可以有效的降低模仿学习对于强化学习的负面影响，同时保证强化学习可以在离线数据集上起作用。

## 5.4 系统测试

为了测试本文提出的基于强化学习的电影推荐系统的泛化效果，本节使其在一个全新的电影数据集 Movielens-1m 上进行推荐。通过在分布不同的数据集上进行推荐系统的性能测试，更能得出具有说服力的结论。

其中 Movielens-1m 数据集的详细信息见表 5-7。从表中可以看出，Movielens-1m 数据集的数据量相比 Movielens-100k 数据集提升了 10 倍多，用户数量提升了 7 倍，电影数量提升了 2 倍多，因此数据集的分布也与 100k 数据集大不相同。

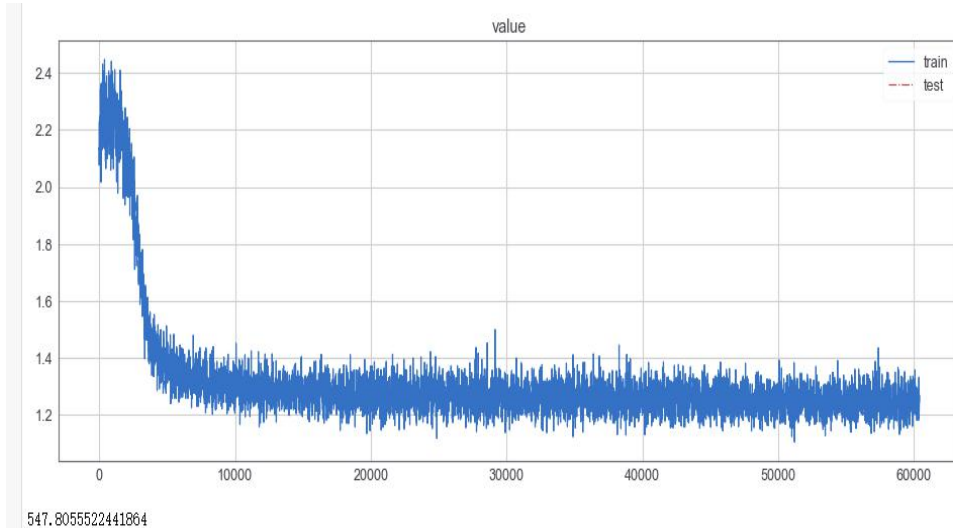
表 5-7 Movielens-100k 数据集信息

Table 5-7 Movielens-1m dataset information

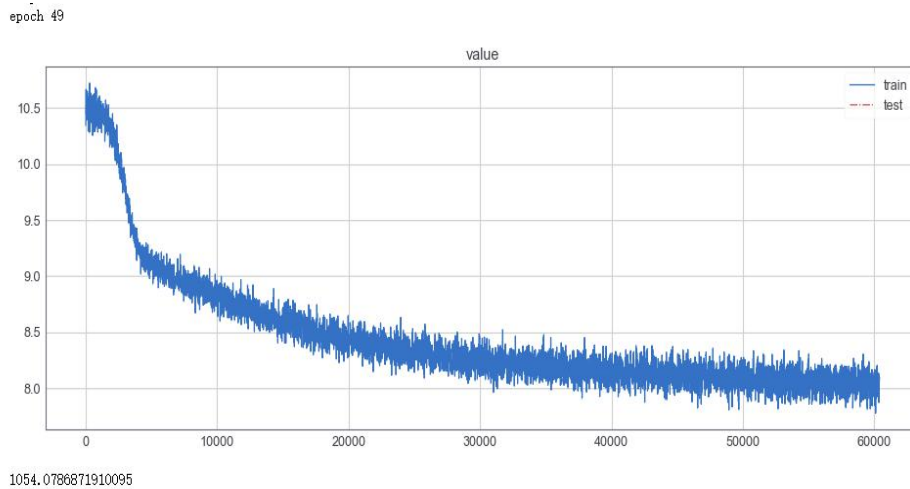
Movielens-1m	
用户数	6040
电影数	3952
交互记录条数	1000209
用户最短交互长度	20
用户最长交互长度	2314
评分范围	1-5

本节的实验环境与上文相同，并通过与上文完全一样的步骤对数据集进行处理得出推荐系统所需的电影嵌入向量以及强化学习素材，然后进行电影推荐，依然是从推荐

结果的命中率来对推荐系统的性能进行分析。算法训练的损失过程见图 5-4。从算法训练的过程来看，将模仿学习加入强化学习的思路与单一的强化学习相比，其在离线数据集上训练的潜力更大。



(a) DQN 网络训练过程



(b) 解耦网络  $\tau=0.9$  训练过程

图 5-4 算法训练的损失下降过程

Fig.5-4 Loss reduction process of algorithm training

本节使用的数据集是 Movielens-1m 数据集，它的数据量相比于原本的 100k 数据集增加了 10 倍，所以本节选用批次训练数量为 5 的训练方式来加快推荐系统的训练速度，并且本节为了验证由上文提出了改进推荐算法方法后推荐系统的效果，只针对 DQN 系列推荐算法进行实验，并增大了推荐范围来更深入地对推荐系统的效果进行分析。详

细实验结果见表 5-8。

表 5-8 测试数据集的推荐效果

Table 5-8 Recommendation effect of test data set

训练轮数 T=50	N=5				
	k=5	k=10	k=20	k=40	k=80
DQN	0.0326	0.0352	0.0420	0.0475	0.0492
I-DQN	0.0534	0.0540	0.0542	0.0534	0.0513
解耦 $\tau=0.3$	0.0457	0.507	0.0574	0.0584	0.0377
解耦 $\tau=0.5$	0.0543	0.0647	0.0707	0.0575	0.0377
解耦 $\tau=0.7$	0.0689	0.0863	0.0695	0.0541	0.0374
解耦 $\tau=0.9$	0.1064	0.1067	0.0655	0.0585	0.0377
解耦 $\tau=1.0$	0.1182	0.1085	0.0631	0.0523	0.0377

因为 1m 数据集中的电影数量为 3000 多部,所以本次实验增加了推荐数量为 40 和 80 两种设置来更加全面的测试推荐系统的效果。同时为了对比解耦网络方法的性能区别,分别测试了  $\tau$  从 0.3 到 1.0 总共中共五种不同设置下的推荐效果。

从表中的实验结果可以分析得出:

1. 普通 DQN 推荐算法与增加了模仿学习后的 I-DQN 推荐算法相比,增加模仿学习后的推荐算法的命中率在所有推荐数量情况下都更高,说明加入模仿学习确实能够有效提高算法的推荐效果。同时普通 DQN 推荐算法的命中率在推荐数量增大的过程中甚至有所上升,而 I-DQN 推荐算法的推荐效果一直很稳定,侧面说明了模仿学习对强化学习状态动作对的选择起到了约束效果。

2. 在  $\tau$  取 0.3 时,解耦网络算法的效果与普通 DQN 推荐算法的效果比较接近,说明  $\tau$  值的选取越接近于 0 时算法性能与普通 DQN 越接近,应证了解耦网络设计的核心思想,即利用  $\tau$  值来控制模仿学习对于强化学习的控制程度。

3. I-DQN 推荐算法与 5 种不同  $\tau$  设置的解耦算法相比,在  $\tau$  取 0.5 以上后,解耦算法的效果在推荐数量低于 80 的情况下要好过损失联合训练的 I-DQN 推荐算法,并且  $\tau$  值选取的越高,低数量推荐的效果越好,  $\tau$  值选取越低,推荐数量增加后的效果更加好,再次应证了  $\tau$  值的设置对于推荐效果控制起到的控制作用。

4. 从 5 种  $\tau$  值在不同推荐数下各自命中率对比,可以看出不同设置的解耦推荐算



法取得最佳推荐效果的推荐数量是不一样的， $\tau$  值与推荐数量呈反比。

5. 与 100k 数据集的命中率相比，推荐系统在 1m 数据集中的命中率总体是下降的，分析原因应该是因为数据集的量变大了，基础推荐系统中的设置不足以对数据集中的数据进行完全的学习，还需要对部分模块做更细节的修改以增强推荐系统在更大数量情况下的整体效果，但是算法改进的方法在新的数据集中依然体现出了应有的效果，验证了本文提出的推荐系统是有效的。

## 5.5 本章小结

本章主要介绍了实验环境以及数据集处理的具体方法，并针对前文提出的三种基础推荐方法进行了有效性实验和性能实验，从实验结果的命中率和训练速度上进行分析，得出各个方法最适合的训练方式。随后针对基于模仿学习改进的 DQN 推荐算法进行多组实验，通过实验结果对改进方法的效果进行验证。最后通过在全新的数据集上进行测试实验来验证整个推荐系统的有效性。

## 结论与展望

### 研究结论

推荐系统是互联网时代的诞生的产物。作为一种解决信息过载问题强有力的工具，可以有效帮助人们寻找需要的信息或商品。以此为背景，本文针对传统的推荐算法只能静态周期性更新而缺乏长效性这一问题，研究了一种基于强化学习的电影推荐系统，采用离线收集的用户交互日志中的数据来对推荐算法进行训练，通过对日志数据进行处理来获取可以用于强化学习推荐算法的训练数据。针对三种经典的强化学习方法，分别实现了三种推荐算法并进行实验。通过实验结果分析研究基于数据集训练的强化学习推荐算法自身所带有的缺陷并提出改进方法，最后通过实验进行验证。论文主要完成工作和结论如下：

（1）完成基于强化学习的电影推荐系统框架的设计。用 Movielens-100k 数据集中的用户交互日志作为素材，完成电影嵌入向量的生成，同时实现了一种日志数据的处理方法，用于生成强化学习推荐算法训练所需要的强化学习四元组数据。

（2）针对经典的强化学习算法进行修改，分别实现了基于 DQN 的推荐算法、基于 DDPG 的推荐算法以及基于 Reinforce 的推荐算法，并通过实验对三种推荐算法的效果进行了对比分析。同时由 DDPG 和 DQN 推荐算法的性能问题切入，分析了以数据集训练的强化学习推荐算法出现误差的来源。

（3）为了消除训练的误差来源，提出一种将模仿学习与强化学习相结合的推荐算法，通过在训练过程中将状态与动作进行约束来限制强化学习产生误差。经过实验对比，证明了新的推荐算法的效果。

（4）针对模仿学习与强化学习一起训练存在的推荐不稳定不可控的问题，提出一种解耦网络的改进训练方法，通过对强化学习和模仿学习分别建立网络来将训练过程解绑，并使用一个参数  $\tau$  来调节模仿学习对于强化学习的控制程度。经过实验对比，解耦网络改进方法取得实际效果。

（5）最后通过在全新数据集 Movielens-1m 上再次实验来测试本文提出的推荐系统的效果，最终验证了本文所提出的推荐系统以及改进训练方法的有效性。

## 未来研究展望

强化学习作为一种模拟人类探索交互的学习方式,在条件成熟的情况下将会被广泛地应用于各类决策问题当中。本文研究的是一种基于数据集训练的强化学习推荐系统,针对如何在推荐系统这一领域使用强化学习方法进行了初步的尝试和探索,对实验中已经出现的问题进行了细致的分析,但由于目前公开的资料有限,所以基础工作所消耗的时间过长,未能更深入的对问题进行研究。后续可以从以下几个方面进行改进:

(1) 在推荐系统领域还有许多不同类型网络结构框架,本文中只使用了最简单的三层神经网络结构来对用户的状态进行训练,后续可以有针对性的使用不同的网络结构来测试算法的性能。

(2) 本文电影的嵌入训练采用了最简单的 Word2vec 模型进行训练,之后可以尝试使用最近几年出现的图嵌入模型对物品作嵌入来提高嵌入的准确性。

(3) 使用数据集数据训练强化学习这方面的研究可以继续推进。

(4) 目前强化学习推荐算法只能应用于物品数量不大的推荐系统,后续需要对较大物品数量的应用场景进行研究和修改。

---

## 参考文献

- [1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions[J]. IEEE Transactions on Knowledge and Data Engineering ,2005,734 - 749.
- [2] Gediminas Adomavicius, Nikos Manouselis, and YoungOk Kwon. Multicriteria recommender systems.In Recommender Systems Handbook[J]. Boston, MA:Springer, 2011,769 - 803.
- [3] 项亮. 推荐系统实践[M]. 人民邮电出版社, 2012.
- [4] 刘建国, 周涛, 郭强等. 个性化推荐系统评价方法综述[J]. 复杂系统与复杂性科学, 2009, 6(3):1-10.
- [5] Zhou T, Kuscsik Z, Liu J G, et al. Solving the apparent diversity-accuracy dilemma of recommender systems[J]. Proceedings of the National Academy of Sciences,2010, 107(10):4511-4515.
- [6] Mcnee S M, Riedl J, Konstan J A. Being accurate is not enough: how accuracy metrics have hurt recommender systems[J]. Proc Chi, 2006:1097-1101.
- [7] C. A. Gomez-Uribe, N. Hunt. The netflix recommender system: Algorithms, business value, andinnovation[J]. ACM Transactions on Management Information Systems (TMIS), 2016, 6(4):13.
- [8] R. Catherine, W. Cohen. Transnets: Learning to transform for recommendation[C]. Proceedings of the Eleventh ACM Conference on Recommender Systems, 2017, 288-296.
- [9] P. Resnick, H. R. Varian. Recommender systems[J]. Communications of the ACM, 1997, 40(3):56-59.
- [10] G. Adomavicius, A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions[J]. IEEE Transactions on Knowledge & Data Engineering, 2005, 17(6): 734-749.
- [11] W. Zhou, J. Li, M. Zhang, et al. Incorporating social network and user's preference in

- matrixfactorization for recommendation[J]. Arabian Journal for Science and Engineering, 2018, 43(12):8179-8193.
- [12] Resnick P, Iacovou N, Suchak M, et al. GroupLens: an open architecture for collaborative filtering of netnews[J]. conference on computer supported cooperative work, 1994: 175-186.
- [13] Linden G, Smith B R, York J C, et al. Amazon.com recommendations: item-to-item collaborative filtering[J]. IEEE Internet Computing, 2003, 7(1): 76-80.
- [14] Breese J S, Heckerman D, Kadie C M, et al. Empirical analysis of predictive algorithms for collaborative filtering[J]. uncertainty in artificial intelligence, 1998: 43-52.
- [15] Pazzani M J. A Framework for Collaborative, Content-Based and Demographic Filtering[J]. Artificial Intelligence Review, 1999, 13(5): 393-408.
- [16] T. Ebesu, Y. Fang. Neural citation network for context-aware citation recommendation[C]. Pro-ceedings of the 40th International ACM SIGIR Conference on Research and Development inInformation Retrieval, 2017, 1093-1096.
- [17] B. Sarwar, G. Karypis, J. Konstan, et al. Item-based collaborative filtering recommendation algo-rithms[C]. Proceedings of the 10th International Conference on World Wide Web, 2001, 285-295.
- [18] R. Salakhutdinov, A. Mnih. Probabilistic matrix factorization[C]. Advances in Neural Informa-tion Processing Systems, 2008, 1257-1264.
- [19] R. Salakhutdinov, A. Mnih. Bayesian probabilistic matrix factorization using markov chain montecarlo[C]. Proceedings of the 25th International Conference on Machine Learning, 2008, 880-887.
- [20] P. Massa, P. Avesani. Trust-aware recommender systems[C]. Proceedings of the 2007 ACM Con-ference on Recommender Systems, 2007, 17-24
- [21] Choi S, Ha H, Hwang U, et al. Reinforcement Learning based Recommender System using Biclustering Technique[J]. ResearchGate, 2018: 179-187.
- [22] Zheng G J, Zhang F Z, Zheng Z H, et al, DRN: A Deep Reinforcement Learning

- Framework for News Recommendation[C]. WWW 2018: 167-176.
- [23] Sutton R S, Barto A G. Reinforcement Learning: An Introduction[M]. Neural information processing systems, 1999.
- [24] Q. Liu, E. Chen, H. Xiong, et al. Enhancing collaborative filtering by user interest expansion via personalized ranking[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2012,42(1): 218-233.
- [25] Q. Song, J. Cheng, H. Lu. Incremental matrix factorization via feature space re-learning for recommender system[C]. Proceedings of the 9th ACM Conference on Recommender Systems, 2015,277-280.
- [26] A. Salah, N. Rogovschi, M. Nadif. A dynamic collaborative filtering system via a weighted clustering approach[J]. Neurocomputing, 2016, 175: 206-215.
- [27] M. Jiang, P. Cui, F. Wang, et al. Scalable recommendation with social contextual information[J]. IEEE Transactions on Knowledge and Data Engineering, 2014, 26(11): 2789-2802.
- [28] G. Zhou, X. Zhu, C. Song, et al. Deep interest network for click-through rate prediction[C]. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, 1059-1068.
- [29] D. M. Blei, A. Y. Ng, M. I. Jordan. Latent dirichlet allocation[J]. J Machine Learning Research Archive, 2003, 3: 993-1022.
- [30] S. Zhang, W. Wang, J. Ford, et al. Learning from incomplete ratings using non-negative matrix factorization[C]. Proceedings of the 2006 SIAM international conference on data mining, MD, USA, 2006, 549 – 553.
- [31] R. Salakhutdinov, A. Mnih, G. Hinton. Restricted boltzmann machines for collaborative filtering[C]. Proceedings of the 24th International Conference on Machine Learning, 2007, 791-798.
- [32] S. Zhang, W. Wang, J. Ford, et al. Learning from incomplete ratings using non-negative matrix factorization[C]. Proceedings of the 2006 SIAM international conference on data mining, MD, USA, 2006, 549 – 553.

- [33] F. Strub, R. Gaudel, J. Mary. Hybrid recommender system based on autoencoders[C]. Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, 2016, 11-16.
- [34] B. Sarwar, G. Karypis, J. Konstan, et al. Item-based collaborative filtering recommendation algorithms[C]. Proceedings of the 10th International Conference on World Wide Web, 2001, 285-295.
- [35] G. Linden, B. Smith, J. York. Amazon.com recommendations: Item-to-item collaborative filtering[J]. IEEE Internet Computing, 2003, 7: 76-80.
- [36] A. Salah, N. Rogovschi, M. Nadif. A dynamic collaborative filtering system via a weighted clustering approach[J]. Neurocomputing, 2016, 175: 206-215.
- [37] S. Kabbur, X. Ning, G. Karypis. Fism: factored item similarity models for top-n recommender systems[C]. Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2013, 659-667.
- [38] H.-T. Cheng, L. Koc, J. Harmsen, et al. Wide & deep learning for recommender systems[C]. Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, 2016, 7-10.
- [39] M. Polato, F. Aioli. Boolean kernels for collaborative filtering in top-n item recommendation[J]. Neurocomputing, 2018, 286: 214-225.
- [40] A. L. V. Pereira, E. R. Hruschka. Simultaneous co-clustering and learning to address the cold start problem in recommender systems[J]. Knowledge-Based Systems, 2015, 82: 11-19.
- [41] A. Salah, N. Rogovschi, M. Nadif. A dynamic collaborative filtering system via a weighted clustering approach[J]. Neurocomputing, 2016, 175: 206-215.
- [42] T. Ebesu, Y. Fang. Neural citation network for context-aware citation recommendation[C]. Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2017, 1093-1096.
- [43] B. Carminati, E. Ferrari, J. Girardi. Trust and share: Trusted information sharing in online social networks[C]. IEEE 28th International Conference on Data Engineering,

- 2012, 1281-1284.
- [44] F. Liu, J. L. Hong. Use of social network information to enhance collaborative filtering performance[J]. Expert Systems with Applications, 2010, 37(7): 4772-4778.
- [45] J. Wei, J. He, K. Chen, et al. Collaborative filtering and deep learning based recommendation system for cold start items[J]. Expert Systems with Applications, 2017, 69: 29-39.
- [46] L. Song, C. Tekin, M. V. D. Schaar. Online learning in large-scale contextual recommender systems[J]. IEEE Transactions on Services Computing, 2016, 9(3): 433-445.
- [47] F. Yuan, G. Guo, J. M. Jose, et al. Optimizing factorization machines for top-n context-aware recommendations[C]. International Conference on Web Information Systems Engineering, 2016, 278-293.
- [48] D. M. Blei, A. Y. Ng, M. I. Jordan. Latent dirichlet allocation[J]. J Machine Learning Research Archive, 2003, 3: 993-1022.
- [49] S. Fujimoto, D. Meger, D. Precup. Off-policy deep reinforcement learning without exploration[J]. arXiv preprint arXiv:1812.02900 (2018)
- [50] H. Van Hasselt, A. Guez, D. Silver. Deep Reinforcement Learning with Double Q-Learning[C]. AAAI, Phoenix, AZ, 2016, 5.
- [51] G. Dulac-Arnold, R. Evans, H. van Hasselt, et al. Deep reinforcement learning in large discrete action spaces[J]. arXiv preprint arXiv:1512.07679, 2015.
- [52] 王喆. 深度学习推荐系统[M]. 电子工业出版社, 2020.



## 攻读学位期间取得与学位论文相关的成果

发表和投稿与学位论文相关学术论文

## 学位论文独创性声明

本人郑重声明：所呈交的学位论文是我个人在导师的指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明，并表示了谢意。本人依法享有和承担由此论文所产生的权利和责任。

论文作者签名：李翔宇 日期：2021.5.25

## 学位论文授权使用授权声明

本学位论文作者完全了解学校有关保存、使用学位论文的规定：“研究生在广东工业大学学习和工作期间参与广东工业大学研究项目或承担广东工业大学安排的任务所完成的发明创造及其他技术成果，除另有协议外，归广东工业大学享有或特有”。同意授权广东工业大学保留并向国家有关部门或机构送交该论文的印刷本和电子版本，允许该论文被查阅和借阅。同意授权广东工业大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印、扫描或数字化等其他复制手段保存和汇编本学位论文。保密论文在解密后遵守此规定。

论文作者签名：李翔宇 日期：2021.5.25

指导教师签名：王沛 日期：2021.05.25

## 致 谢

回首往事，三年的研究生生活还历历在目，从刚进入实验室作为参加机器人小车比赛的培训员，到后来前往新加坡国立大学进行访问学习，以及回国后遇到疫情半年在家的等待，这一切都似乎才刚刚过去不久。

一路走来，也收到了许多人的帮助。感谢我的导师陈玮教授，对我的悉心栽培，教了我许多人生的道理，并且信任我让我来对实验室的各项事务进行管理。感谢实验室同级同学丘庆、刘岱远、刘奕在日常生活中的合作、陪伴以及插科打诨，让生活中的烦恼减少了许多。感谢郭靖和曹志广师兄在学习上的悉心教导以及指点。感谢我的家人们在我烦恼和疑惑时对我的关心和包容，鼓励我去做尝试。最后感谢广东工业大学对我七年的教育，希望以后可以回报社会，回报学校。

