题目： 基于 GRU 的主动学习协同过滤算法研究

院（系）	计算机科学与技术学院
专　　业	计算机科学与技术专业
届　　别	2018 届
学　　号	1425111035
姓　　名	潘傲寒
指导老师	王成

华侨大学教务处印制

2018 年 5 月

摘　要

推荐算法旨在帮助互联网用户缓解信息过载问题，而协同过滤是其中最重要的一种推荐算法集合。本文研究了基于 GRU 的协同过滤推荐算法，并利用主动学习动态采样数据，主动选择合适的训练集，使得模型能够快速建立。通过在公开数据集 MovieLens 上的实验，验证了本文提出的算法能够有效提升精度，发掘长尾物品。

本文的主要贡献有：

（1）本文分析了 GRU 模型的结构和各种参数对模型的影响。并且通过实验比较，验证了它确实优于其它算法。

（2）本文提出了使用主动学习改进基于 GRU 的协同过滤算法，通过改进 MinRating 算法以适应 TopN 推荐，最终提出基于 GRU 的主动学习协同过滤推荐算法。实验证明，有效加快了模型的训练过程。

关键词：协同过滤；GRU；主动学习

ABSTRACT

Recommender algorithm help Web users to address information overload, and collaborative filtering is one of the most important recommender algorithm. In this paper, we propose the collaborative filtering algorithm based on GRU, meanwhile, active learning is used for dynamic sampling data and selecting the appropriate training set actively so that the model can be built quickly. Experiments on the open dataset MovieLens verify that the algorithm proposed in this paper can effectively improve the accuracy and discover the long tail items.

The main contributions of this paper are:

(1) This paper analyzes the structure of GRU model and the influence of various parameters on the model. And through experimental comparison, it is verified that it is better than other algorithms.

(2) In this paper, active learning is proposed to improve the collaborative filtering algorithm based GRU , and the MinRating algorithm is improved to adapt to the TopN recommendation. Finally, we proposed the active learning collaborative filtering recommendation algorithm based on GRU. Experiments prove that the training process of the model is effectively accelerated.

Keywords: collaborative filtering, GRU, active learning

目 录

第 1 章 绪论

1.1 研究背景及意义

在互联网时代，海量数据信息已经超过了人类能够接受的范围。面对信息过载的问题，有许多的解决方案。目前最广泛应用的是搜索引擎和推荐系统，搜索引擎通过给定若干个关键词，使用信息检索算法，找到相关的信息，而当用户没有关键词时，则使用不了搜索引擎。推荐系统不需要用户提供关键词，而是主动分析用户的历史数据，主动给用户提供内容服务，解决了搜索引擎在没有明确意图场景下无法使用的问题。

随着互联网的发展，商品推荐、视频推荐、音乐推荐、新闻推荐等各种个性化推荐应用走上舞台。推荐算法支撑着这些应用，而协同过滤推荐算法是其中最成功的算法之一。

1.2 协同过滤推荐算法定义

协同过滤算法的本质是，对系统内用户过去的所有行为数据建模，预测用户对系统内项目的兴趣程度，从而实现推荐任务。

设用户集合为 $U = \{u_1, u_2, \dots, u_N\}$，项目集合 $I = \{i_1, i_2, \dots, i_N\}$。用户对项目的历史交互行为集合 R，例如评分行为。映射关系为：

$$U \times I \rightarrow R \tag{1.1}$$

推荐系统的目的，就是寻找最优的映射函数，对于用户 u 对项目 i 的兴趣程度记为 P(u, i)。对于任意一个用户 $u \in U$，通过 P 为用户 u 推荐 i，公式如下：

$$i^* = \arg\max P(u, i) \tag{1.2}$$

协同过滤推荐算法的目的就是建立模型 P，利用用户和项目的历史交互行为来最大化函数 P。

1.3 协同过滤研究现状

根据推荐规则和数据的不同，可以将推荐系统分为以下几类：基于内容的推荐算法、协同过滤推荐算法、基于规则的推荐算法、基于矩阵分解的推荐算法等[1]。基于内容的推荐算法通过提取物品内容特征，例如标签、文本内容，生成特征向量，在文本物品种能发挥很好的作用，例如个性化新闻推荐；协同过滤是利用用户群和物品集合之间的行为数据进行推荐，传统的方法是计算用户或物品之间的相似性，然后利用相似用户或相似物品计算用户对物品的喜好程度，从而将物品推荐给用户；基于矩阵分解的算法，又称隐语义模型，常被归纳于协同过滤推荐算法中。本质上是降维，因为精度高，速度快，是目前的主流方法。

1.3.1 基于邻域

基于领域的协同过滤推荐算法，利用领域相似性，计算 n 个相似的邻居，通过邻居来推荐。基于邻域分为基于用户和基于项目两种。

基于用户：1）计算 n 个相似的用户；2）将 n 个用户喜好的项目推荐给目标用户。

基于项目：1）计算项目之间的相似度；2）将目标用户喜好项目的 n 个近似项目推荐给目标用户。

以基于用户为例，分为相似度计算阶段和兴趣度计算阶段。

（1）相似度计算

隐性反馈行为，指那些中立的用户项目交互行为，只能表示用户有交互，而不代表用户的喜好。例如页面浏览行为。隐性反馈行为数据大量存在，是实际生产环境中最多的数据。一个用户对一个物品只有两种状态，有交互和无交互。如图 1.1，图中用户有对物品交互行为则标为 1，未交互则标为问号。

	物品1	物品2	物品3	物品4
用户1	1	?	1	?
用户2	1	?	1	1
用户3	?	1	1	1

图 1.1　隐性反馈行为矩阵

令 N(u)表示行为矩阵中，第 u 行所有为 1 的列；N(v)表示第 v 行所有为 1

的列。Jaccard 公式：

$$w_{uv}=\frac{|N(u)\cap N(v)|}{|N(u)\cup N(v)|} \tag{1.3}$$

余弦公式：

$$w_{uv}=\frac{|N(u)\cap N(v)|}{\sqrt{|N(u)||N(v)|}} \tag{1.4}$$

对于显性反馈行为，指带有情感倾向的交互行为。例如电影评分行为。常见的是很多网站使用的 5 分评级系统。movielens 数据集就属于显性反馈。如图 1.2，图中用户对项目评分为 1 到 5 的整数，未评分则为问号。

	物品1	物品2	物品3	物品4
用户1	4	2	1	?
用户2	3	1	1	5
用户3	?	4	?	4

图 1.2　显性反馈行为矩阵

令 S(u,v)为矩阵中第 u 行和第 v 行同为非问号的列，r_{xy} 为第 x 行第 y 列的值，余弦公式：

$$w_{uv}=\frac{\sum_{i\in S(u,v)} r_{ui}r_{vi}}{\sqrt{\sum_{i\in S(u,v)} r_{ui}^2 \sum_{i\in S(u,v)} r_{vi}^2}} \tag{1.5}$$

皮尔逊公式：

$$w_{uv}=\frac{\sum_{i\in S(u,v)} (r_{ui}-\overline{r_u})(r_{vi}-\overline{r_v})}{\sqrt{\sum_{i\in S(u,v)} (r_{ui}-\overline{r_u})^2 \sum_{i\in S(u,v)} (r_{vi}-\overline{r_v})^2}} \tag{1.6}$$

（2）兴趣度计算

使用公式 1.7，计算用户对项目的兴趣程度。T(u,K)代表使用近似公式计算出的 K 个最大近似度的用户，N(i)是评分矩阵中第 i 列不为问号的行。

$$p(u,i)=\sum_{v\in T(u,K)\cap N(i)} w_{uv}r_{vi} \tag{1.7}$$

1.3.2 基于矩阵分解

用户-项目交互行为，例如电影评分，可以组织成一个用户-电影评分矩阵。基于矩阵分解的推荐算法，将评分矩阵分解为若干个子矩阵。本质上是通过降维补全评分矩阵，从而得到电影的评分预测。

最早的协同过滤矩阵分解，使用的是奇异值分解 SVD[2]。对于一个评分矩阵 $R \in \mathbb{R}^{m \times n}$，第一步是将评分矩阵补全，通常使用的是平均值。但是评分矩阵往往极大且稀疏，难以存储，SVD 效率很低，并且补全后精度很低。

2006 年 Netflix Prize 开始后，Simon Funk 在自己的博客中公布了 SVD 的改进算法，被称为 Funk-SVD，又被称为 Latent Factor Model，它将评分矩阵分解为两个低维矩阵相乘[3]。

$$\hat{R} = P^T Q \tag{1.8}$$

其中 $P \in \mathbb{R}^{f \times m}$，$Q \in \mathbb{R}^{f \times n}$。P 是 m 个用户的 f 维特征，Q 是 n 个物品的 f 维特征，可由公式 1.9 得到评分预测值。

$$\hat{r}_{ui} = \sum_f p_{uf} q_{if} \tag{1.9}$$

损失函数为，其中 λ 是正则化参数：

$$C(p,q) = \sum_{(u,i) \in Train} (r_{ui} - \sum_{f=1}^{F} p_{uf} q_{if})^2 + \lambda(\| p_u \|^2 + \| q_i \|^2) \tag{1.10}$$

训练模型时，使用迭代算法，例如随机梯度下降，调整 PQ 矩阵，最小化损失函数。使用模型时，利用公式 1.10 计算预测评分值。

Koren 提出将用户历史评分的物品加入 LFM 模型中，称作 SVD++模型[4]。模型公式如下：

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \cdot (p_u + \frac{1}{\sqrt{|N(u)|}} \sum_{j \in N(u)} y_j) \tag{1.11}$$

其中，μ 是训练集中所有评分的平均数，通过预先计算得到；b_u 是用户偏置项，b_i 是项目偏置项，是通过训练获得的参数。N(u)是用户 u 评价的项目集合。y_i 是参数，通过训练获得。

1.3.3 基于深度学习

国内外研究深度学习应用至协同过滤推荐系统的成果中，大多数都在改进基于矩阵分解的方法，使用深度学习模型做矩阵分解。

Neural Collaborative Filtering 模型[5]，提出了一个通用框架来对协同过滤数据进行矩阵分解，如图 1.3 所示。NCF 的输入是项目和用户的 one-hot 向量。嵌入层（Embedding Layer）将稀疏的 one-hot 向量映射成 embedding 向量，然后输入多个全连接层，最终预测出评分值。

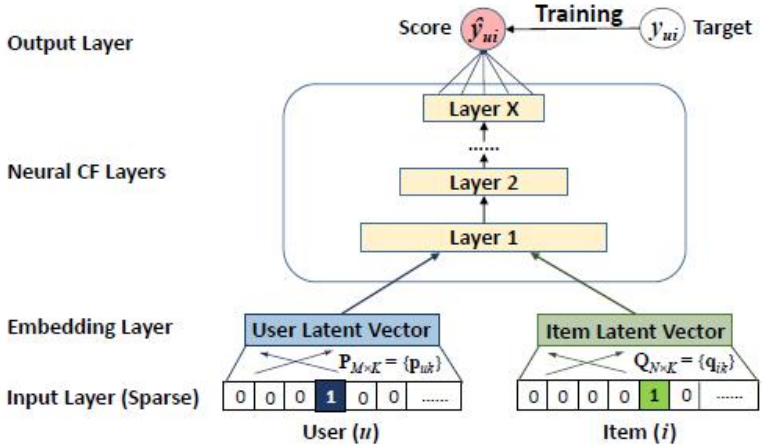

图 1.3　NCF 深度学习模型

Deep Matrix Factorization 模型[6]，提出直接通过神经网络结构，将用户和项目投射到潜在空间中的低维向量中，如图 1.4 所示。DMF 中有两个多层全连接模块，输入分别是评分矩阵行和列（用户对所有物品评分的情况和物品被所有用户评分的情况），两个输入向量分别通过多层全连接模块，最终得到两个隐层语义向量，使用余弦公式度量两个向量的近似程度，近似程度即为用户对该物品的喜好预测值。

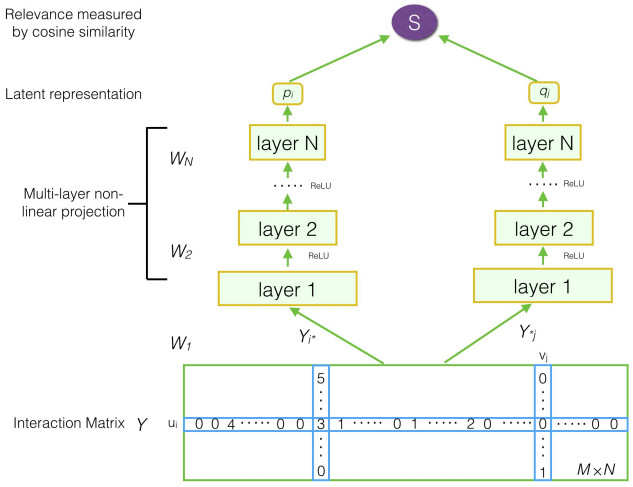

图 1.4　DMF 深度学习模型

1.4 论文的组织结构

本文针对协同过滤推荐算法，围绕着提升指标精度和发掘长尾物品的能力，提出了基于 GRU 的主动学习协同过滤推荐算法。本文分为四个章节，各章节安排如下：

第一章：绪论。探讨了推荐算法的研究背景及意义，引入协同过滤算法的定义，从基于邻域、基于矩阵分解和基于深度学习的角度简介了协同过滤算法的研究现状。最后介绍了各章节的内容安排。

第二章：基于 GRU 的协同过滤推荐算法。提出将协同过滤数据转换成时序数据，使用动态的数据环境。第二部分介绍了循环神经网络，提出使用 GRU 建模时序化后的协同过滤数据。最后提出协同过滤的几个关键指标，通过实验，探讨了隐层参数维度大小、迭代优化算法的选择、Dropout 概率和 TopN 对 GRU 的影响，并与多种方法进行比较，最后得出结论。

第三章：基于主动学习的改进。改进 MinRating 主动学习算法，以适应 TopN 推荐。提出基于 GRU 的主动学习协同过滤推荐算法。

第四章：总结与展望。梳理本文提出的“基于 GRU 的主动学习协同过滤算法”，阐述算法的优点，存在的缺陷和进一步改进的空间。

第 2 章 基于 GRU 的协同过滤推荐算法

协同过滤问题可以被看作是一个序列预测问题，在这种情况下，循环神经网络是一个非常有竞争力的方法。本章研究了门控单元（GRU）如何应用于协同过滤，并将它和传统的基于邻域和基于矩阵分解的方法以及时序建模方法马尔科夫模型在电影推荐中比较。

2.1 时序化

在大部分协同过滤推荐的公开数据集中，是由多条用户-物品-评分-时间戳数据组成的集合；而在实际的生产环境中，后台日志系统会自动清洗出带时间戳的隐性反馈数据，数据库中会保存带时间戳的显性反馈数据。在用户与物品交互过程中，何时交互这是一个很重要的信息。用户 U 在评价物品 A 后，评价了物品 B，这里隐含了用户的兴趣转移，物品 A 和物品 B 的转移信息没有被传统的协同过滤算法捕捉到。利用时间戳，将评分矩阵转换成时间序列数据，既充分利用了评分时间信息，并且为深度学习模型循环神经网络提供了可用的数据格式。

在传统的 TopN 推荐任务中，用户 i 在时刻 t，S_i^{t-}是用户 i 在 t 时刻前交互的物品集合，S_i^{t+}是用户 i 在 t 时刻后交互的物品集合。推荐系统的目标就是提供一个 S_i^{t-}映射到 S_i^{t+}的函数，在这里的环境是静态的，项目交互的顺序与推荐算法无关。将环境改成动态，推荐算法不仅仅基于用户交互的项目，还基于这个用户交互项目的顺序。用户的交互行为由一个交互序列表示：他依次交互了 x1, x2, x3。推荐算法的目标是根据开始的三个项目组成的序列，预测接下来的交互行为（x4, x5, ...）。将协同过滤问题变治为一个序列预测问题，会产生短期和长期预测两个概念。短期预测的目的是预测用户下一个将与哪个项目交互（即在最后一个之后），而长期预测的目的是预测最终用户将消耗哪些项目。在静态环境中，这种区分是没有意义的，因为在 S_i^{t+}中的项目的顺序被忽略，并且静态环境中的只有长期预测。然而，在序列预测中，通常是训练模型的的短期预测能力。在 2.4 节中，显示了在以训练 GRU 的短期预测能力为目标的同时，能够同时获得较高的召回率和覆盖率。

在评分数据中，基于邻域、基于矩阵分解等传统协同过滤算法使用的抽象数据概念是评分矩阵。对于一个 $R \in \mathbb{R}^{m \times n}$ 的评分矩阵，用户数 m 和物品数 n 往往很大。物品的数量一般远超过一万，而用户时间精力有限，了解并愿意评分的物品很少超过 100 个，所以导致评分矩阵 R 通常极大且稀疏，这里不讨论如何存储评分矩阵。如图 2.1 所示，问号表示用户未对该物品评分，每个单元格由评分和时间组成。传统协同过滤算法一般会忽略时间信息，直接处理评分数据矩阵。

这里采用的时序化方法，是以用户为单位，按时间升序排序。对于每一个用户，从之前的一行评分数据转换成一条时间序列，如图 2.2 所示。

图 2.1　评分矩阵示例（3 个用户，4 个物品）

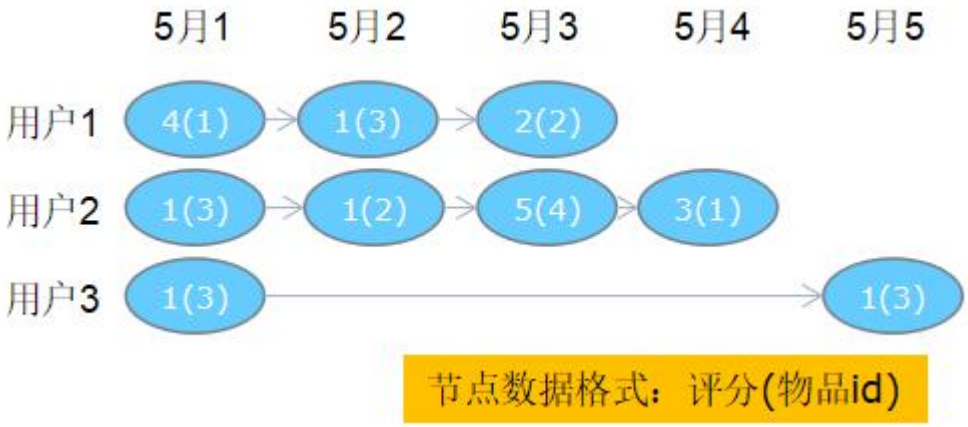

图 2.2　转换后的时序数据

2.2 循环神经网络

循环神经网络（recurrent neural network, RNN）源自 Hopfield 网络[7]。传统的机器学习算法受人工提取的特征影响很大，特征提取得好会很大程度上会提升算法的效果，但是特征提取的研究偏向于经验法，在图像等领域很难提取有效特征。深度学习模型能够自动提取特征，但全连接网络容易过拟合，并且无法对时序数据建模。循环神经网络能够对时序数据建模并且充分挖掘出用数据中深度信息。随着循环神经网络的各种改进和各种变种的提出，时序数据已经能被很好地利用起来。

从网络结构上，循环神经网络能够保留一个内部状态，每一时刻都会对这个内部状态进行更新给下一时刻使用，利用当前时刻的内部状态和输入计算输出值。也就是说，将 RNN 按照时刻展开，两个时刻之间的隐藏层是相互连接的，一个隐藏层的输入包括输入层的值和当前时刻隐藏层的状态。

2.2.1 模型结构

图 2.3 展示了一个标准的循环神经网络。循环神经网络会结合当前时刻的输入和当前时刻的状态，来计算当前的输出。循环神经网络的隐藏层结点 S 的输入会结合输入层的 x_t 和上一时刻保留的状态 s_{t-1}。在每一个时刻，循环神经网络会利用上一时刻的状态计算出当前时刻的状态，并读取 t 时刻的输入 x，输出 o_t。同时 s_t 的状态会被保留着，等待下一时刻使用。因此，循环神经网络实际上是在时刻维度上无限展开的结果。

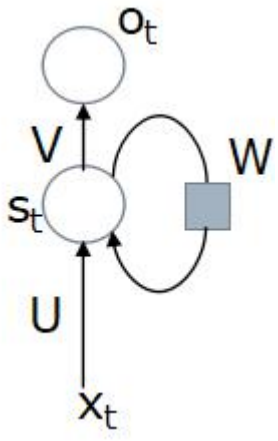

图 2.3　循环神经网络经典结构示意图

实际上，循环神经网络不会无限循环，而是将循环体展开。针对不同问题，不同的数据格式，循环神经网络有不同的展开形式，有一个输入对应一个输出的展开，也有一整个序列对应一个输出的展开，如图 2.4 所示。

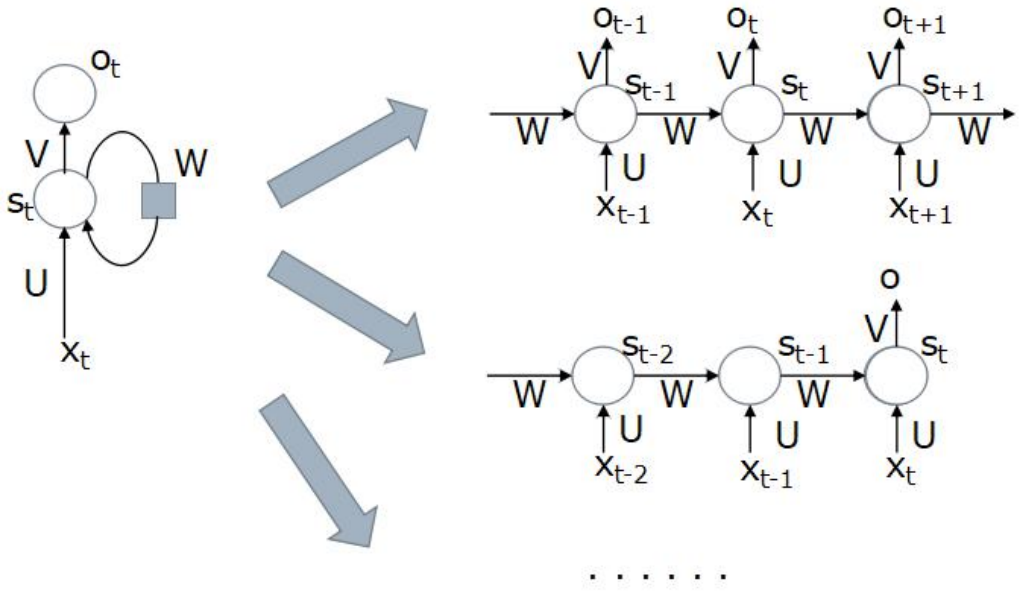

图 2.4　循环神经网络展开形式

循环神经网络有一个隐藏层，一个状态更新公式，一个输出公式。令 X_t 表示 t 时刻的输入，O_t 表示 t 时刻的输出，S_t 表示 t 时刻的隐藏状态。

状态转移公式：

$$S_t = f(UX_t + WS_{t-1}) \tag{2.1}$$

预测/输出公式：

$$O_t = \text{softmax}(VS_t) \tag{2.2}$$

RNN 中的结构细节：

1.隐层始终保留了一个内部状态，这个状态捕捉了之前时间点上的信息。

2.某一时刻的输出，隐含了之前所有输入的信息。

3.隐层状态无法保留之前所有时间点的信息，即使这些信息很重要。随着不断输入，之前时间的信息逐渐被丢失。

4.和卷积神经网络一样，循环神经网络的内部共享参数。如图 2.4 所示，展开后的循环神经网络共享 U、V、W 参数。

5.很多情况下，网络不一定每一时刻都有输出，如图 2.4 所示。因为很多任务，比如序列预测，针对一个序列，只需要最后输出结果即可。

在可以获得 t 时刻前后数据的前提下，一般可以使用 RNN 的改进模型，双向 RNN。

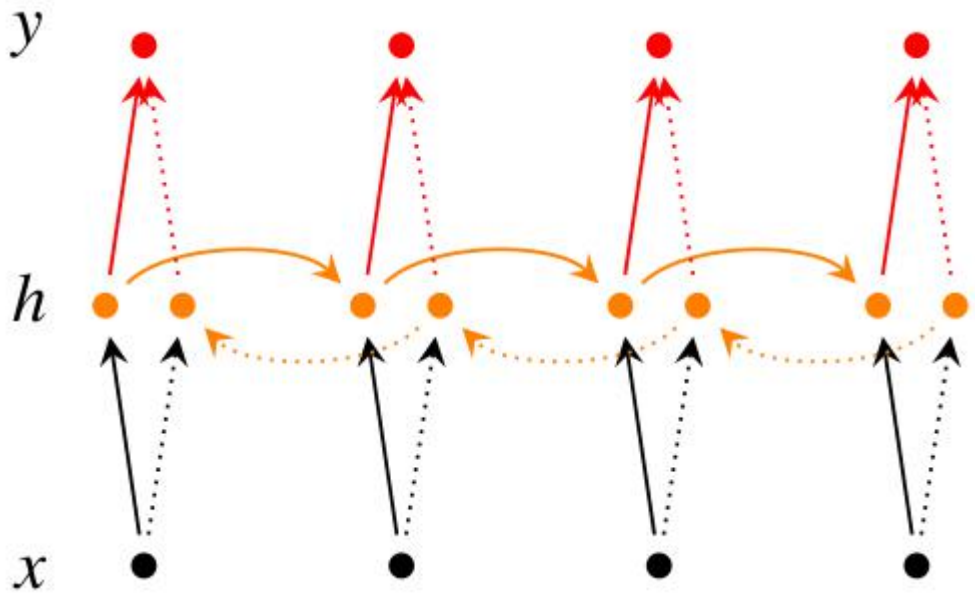

图 2.5　双向 RNN 示意图

从前往后：

$$\vec{S}_t^1 = f(\vec{U}^1 X_t + \vec{W}^1 S_{t-1} + \vec{b}^1) \tag{2.3}$$

从后往前：

$$\vec{S}_t^2 = f(\vec{U}^2 X_t + \vec{W}^2 S_{t-1} + \vec{b}^2) \tag{2.4}$$

输出：

$$o_t = \text{softmax}(V[\vec{S}_t^1; \vec{S}_t^2]) \tag{2.5}$$

这里$[\vec{S}_t^1; \vec{S}_t^2]$将前后两个方向的隐层状态拼接起来，某一时刻的数据同时依赖于前后两个方向的隐层状态。双向 RNN 需要保存两个隐层参数，分别用于向前和向前两个部分，因此双向 RNN 需要占用两倍的内存。在计算某个时刻的输出值时，需要同时输入两个隐藏层输出的信息。

为了记录更多的信息，捕获序列数据中更深层的信息，RNN 可以做进一步的改进，增加隐藏层数。深层双向 RNN 基于这种想法，隐藏层结点输入由上一隐藏层的状态和同层的前一时刻与后一时刻传过来的状态组成。

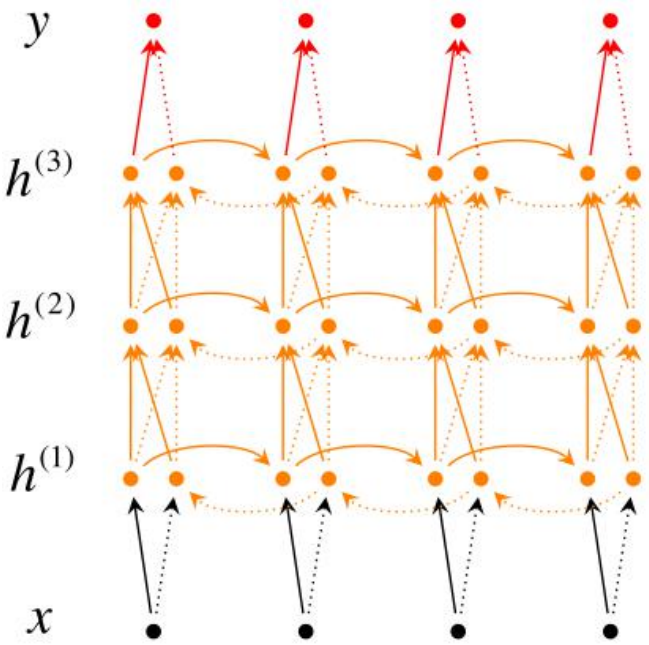

图 2.6　深层双向 RNN 示意图

从前往后：

$$\overrightarrow{h}_t^{(i)} = f(\overrightarrow{W}^{(i)} h_t^{(i-1)} + \overrightarrow{V}^{(i)} \overrightarrow{h}_{t-1}^{(i)} + \overrightarrow{b}^{(i)}) \tag{2.6}$$

从后往前：

$$\overleftarrow{h}_t^{(i)} = f(\overleftarrow{W}^{(i)} h_t^{(i-1)} + \overleftarrow{V}^{(i)} \overleftarrow{h}_{t+1}^{(i)} + \overleftarrow{b}^{(i)}) \tag{2.7}$$

输出层：

$$\hat{y}_t = g(Uh_t + c) = g(U[\overrightarrow{h}_t^{(L)}; \overleftarrow{h}_t^{(L)}] + c) \tag{2.8}$$

RNN 的训练使用的是反向传播，将梯度从最后一个时刻反向传回前一时刻，进行参数的更新训练。这里假设使用的激活函数 f(x) = tanh(x)，那么 t 时刻的隐藏状态为：

$$\mathrm{s_t} = \tanh(Ux_t + Ws_{t-1}) \tag{2.9}$$

对于 softmax 分类器，常使用交叉熵损失函数，t 时刻的损失为：

$$E_t(y_t, \hat{y_t}) = -y_t \log \hat{y_t} \tag{2.10}$$

总损失为每个时刻损失之和：

$$E(y_t, \hat{y_t}) = \sum_t E_t(y_t, \hat{y_t}) = -\sum_t y_t \log \hat{y_t} \tag{2.11}$$

2.2.2 GRU

GRU（Gated Recurrent Unit），是 RNN 的一种变体。GRU 模型比标准 RNN 复杂，比 LSTM 简单，在精度和效率的综合能力上，更适合对时序化后的协同过滤数据建模。

根据 2.2.1 节描述，RNN 网络的模块具有重复性质。标准 RNN 的重复模块结构如图 2.7 所示，它具有参数量小、结构简单等优点。

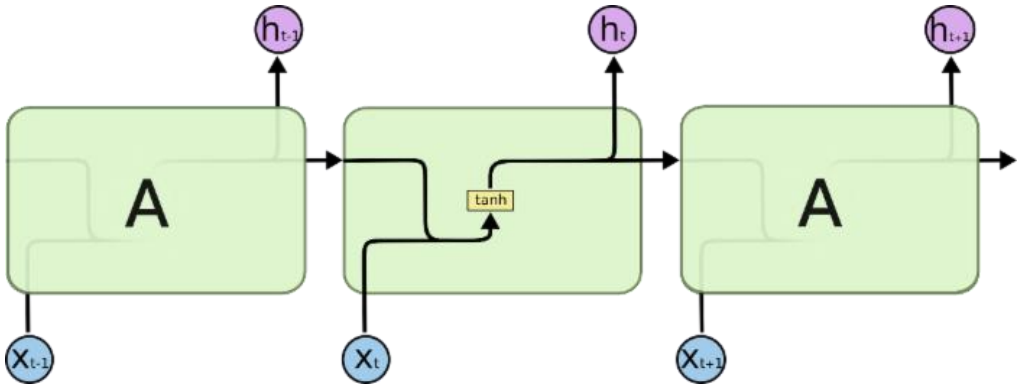

图 2.7 RNN 循环模块

这种模块设计，存在巨大的结构缺陷。随着时间序列的增长，RNN 会很快遗忘掉早先输入的状态。对于一条完整的时间序列数据，RNN 很难学习到大跨度、前后序列位置数据上的潜在关系，因为它很“健忘”。长期依赖是时序数据的特点，但是 RNN 无法解决这个问题。

LSTM（Long Short-Term Memory）是 RNN 的一个变种，能够学习长期依赖关系[8]。标准 LSTM 模型的重复模块如图 2.8 所示。其中，记忆单元，图中的最上面一条黑线，类似于传送带，通过遗忘门和输入门的选择，重要的信息能够在上面保持传送。

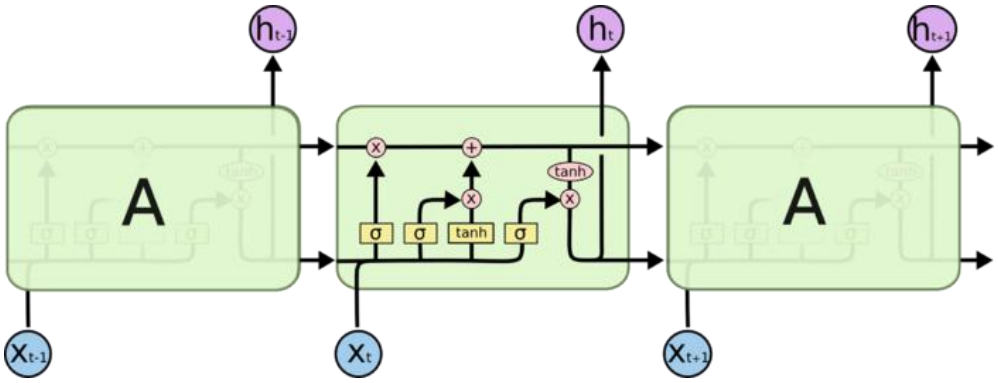

图 2.8 LSTM 循环模块

LSTM 有六个更新公式，两个内部状态，分别为：

遗忘门：$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$ （2.12）

输入门：$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$ （2.13）

新的记忆单元：$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$ （2.14）

最终记忆单元的产生：$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$ （2.15）

输出门：$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$ （2.16）

隐藏状态的更新：$$h_t = o_t * \tanh\left(C_t\right)$$ （2.17）

LSTM 有模型较为复杂，参数的数量较多等问题，所以最终使用的 RNN 模型是 LSTM 的改进版本——GRU。GRU 保留了 LSTM 记忆长期依赖的优点同时又使结构更加简单，易于训练[9]。GRU 的重复模块如图 2.9 所示。

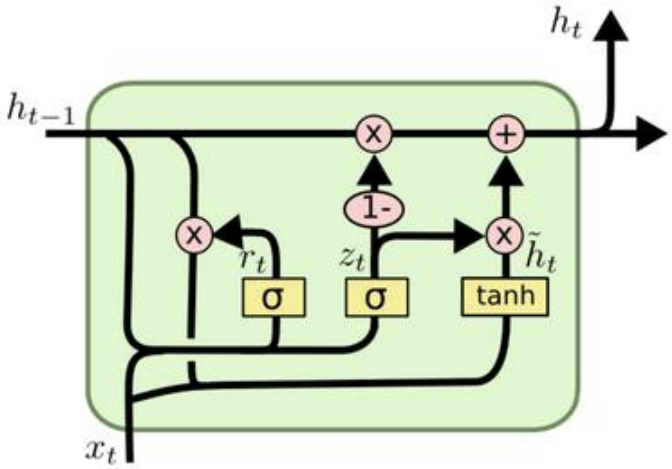

图 2.9 GRU 循环模块

GRU 模型简化了 LSTM 的结构，只有更新门和重置门，即图 2.8 中的 Z_t 和 r_t。更新门用于控制前一时刻的状态信息被带入到当前状态中的程度，更新门的值越大说明前一时刻的状态信息带入越多。重置门用于控制忽略前一时刻的状态信息的程度，重置门的值越小说明忽略得越多。

GRU 有四个更新公式，两个内部状态，分别为：

重置门：$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$ （2.18）

更新门：$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$ （2.19）

新的隐藏状态：$$\tilde{h}_t = \tanh(W_{\tilde{h}} \cdot [r_t * h_{t-1}, x_t])$$ （2.20）

最终隐藏状态的产生：$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$ （2.21）

输出公式为：

$$y_t = \sigma(W_o \cdot h_t) \tag{2.22}$$

GRU 的训练与标准 RNN 相同，使用反向传播算法计算参数的梯度，使用梯度更新算法，例如 adma、adgrad 等，更新参数。从公式 2.18 到 2.22 中，可以看出，需要训练的参数有 W_r、W_z、W_o 和 W_h，前三个参数事拼接的，需要分割出来，即

$$W_r = W_{rx} + W_{rh} \tag{2.23}$$

$$W_z = W_{zx} + W_{zh} \tag{2.24}$$

$$W_{\bar{h}} = W_{\bar{h}x} + W_{\bar{h}h} \tag{2.25}$$

假设输出层的输入 $y_t^i=W_oh$，输出为 $y_t^o= \sigma (y_t^i)$，损失函数为 $E=1/2(y_d - y_t^o)^2$，样本的损失为 $E= \Sigma^T_{t=1}E_t$ ，则参数梯度的求导为：

$$\frac{\partial E}{\partial W_o} = \delta_{y,t} h_t \tag{2.26}$$

$$\frac{\partial E}{\partial W_{zx}} = \delta_{z,t} x_t \tag{2.27}$$

$$\frac{\partial E}{\partial W_{zh}} = \delta_{z,t} h_{t-1} \tag{2.28}$$

$$\frac{\partial E}{\partial W_{\bar{h}z}} = \delta_t x_t \tag{2.29}$$

$$\frac{\partial E}{\partial W_{\bar{h}h}} = \delta_t (r_t \cdot h_{t-1}) \tag{2.30}$$

$$\frac{\partial E}{\partial W_{rx}} = \delta_{r,t} x_t \tag{2.31}$$

$$\frac{\partial E}{\partial W_{rh}} = \delta_{r,t} h_{t-1} \tag{2.32}$$

$$\delta_{y,t} = (y_d - y_t^o) \cdot \sigma' \tag{2.33}$$

$$\delta_{h,t}=\delta_{y,t}W_o+\delta_{z,t+1}W_{zh}+\delta_{t+1}W_{\bar{h}h}\cdot r_{t+1}+\delta_{h,t+1}W_{rh}+\delta_{h,t+1}\cdot(1-z_{t+1}) \tag{2.34}$$

$$\delta_{z,t}=\delta_{t,h}\cdot(h_t-h_{t-1})\cdot\sigma' \tag{2.35}$$

$$\delta_t=\delta_{h,t}\cdot z_t\cdot\phi' \tag{2.36}$$

$$\delta_{r,t}=h_{t-1}\cdot[(\delta_{h,t}\cdot z_t\cdot\phi')W_{\bar{h}h}]\cdot\sigma' \tag{2.37}$$

GRU 输入是协同过滤时序序列，输出维度与项目数相同，代表该用户对每个项目的喜好程度预测，最终选取输出最高的 N 个项目作为 TopN 推荐的结果。

2.3 算法流程

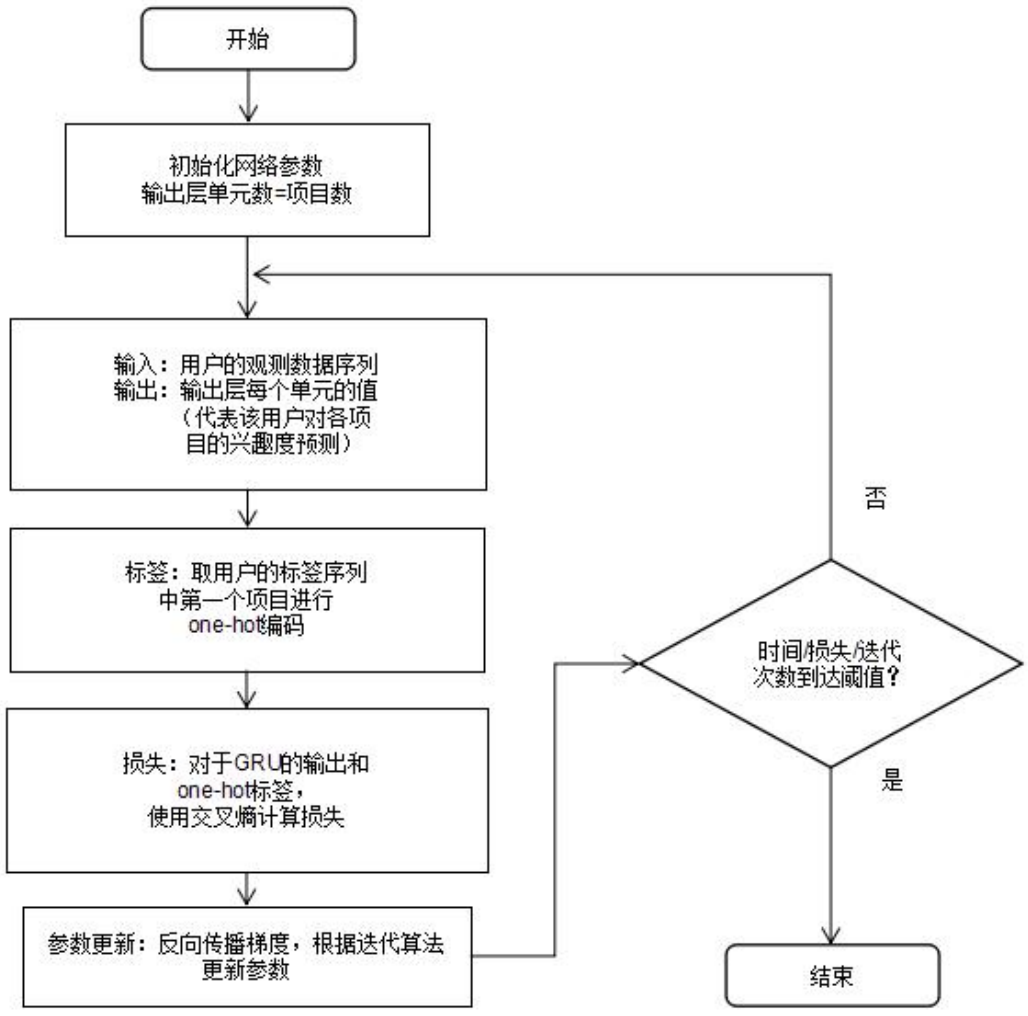

图 2.10 基于 GRU 的协同过滤算法训练流程图

基于 GRU 的协同过滤算法模型，训练过程如图 2.10 所示，网络的输出层单元个数为系统的项目总数，每一个单元对应着相同 id 项目的兴趣度预测。输入用户的观测序列，在 GRU 的输出层中获取对每个项目的兴趣度预测值，经 softmax 归一化，兴趣度是 0 到 1 的概率。标签只使用该用户标签序列的第一个项目的 one-hot 向量，使用交叉熵计算损失，通过反向传播梯度至网络中每个参数，根据迭代算法进行参数的更新，迭代算法的讨论详见 2.4.5 节。

在测试和部署阶段，输入用户的协同过滤序列，在输出层得到对每个项目的兴趣度概率预测。取最大的 N 个单元对应的项目，组合成推荐列表作为算法的推荐结果。N 对算法的影响详见 2.4.7 节。

2.4 理论分析和实验验证

2.4.1 数据集

实验使用数据集是 MovieLens 1M。它是公开的电影评分数据集，其中包含 1000209 条评分数据；6040 名用户，用户 id 从 1 到 6040；3883 部电影，电影 id 从 1 到 3952，即 69 部电影被保留了 id 但是被剔除出数据集，而有被评分的电影 3706 部。数据集保证了每名用户至少留下 20 个评分。数据具体分为三个部分，用户数据、电影数据和评分数据。

（1）用户数据

用户数据由用户（UserID）、性别（Gender）、年龄（Age）、职业（Occupation）、邮编（Zip-code）五个字段组成。。用户性别由 M 对应男士和 F 对应女士标明。年龄分成七组，分别是“小于 18 岁”、“18 到 24 岁”、“25 到 34 岁”、“35 到 44 岁”、“45 到 49 岁”、“50 到 55 岁”和“大于 56 岁”。职业有 21 类，分别为“其它”、“学者”、“艺术家”、“行政人员”、“高校学生/毕业生”、“售后”、“医生”、“管理岗”、“农民”、“主妇”、“K-12 学生”、“律师”、“程序员”、“退休人员”、“销售/市场”、“科学家”、“自由职业”、“技工/工程师”、“商人/工匠”、“失业”和“作家”。

年龄分布如图 2.9 所示，MovieLens 1M 数据集中评分用户主要集中在 20 到 30 岁中间，电影评分偏向年轻人的兴趣爱好。职业分布如图 2.10 所示，用户群题最大的是学生，新潮、商业电影可能有更多的评分。

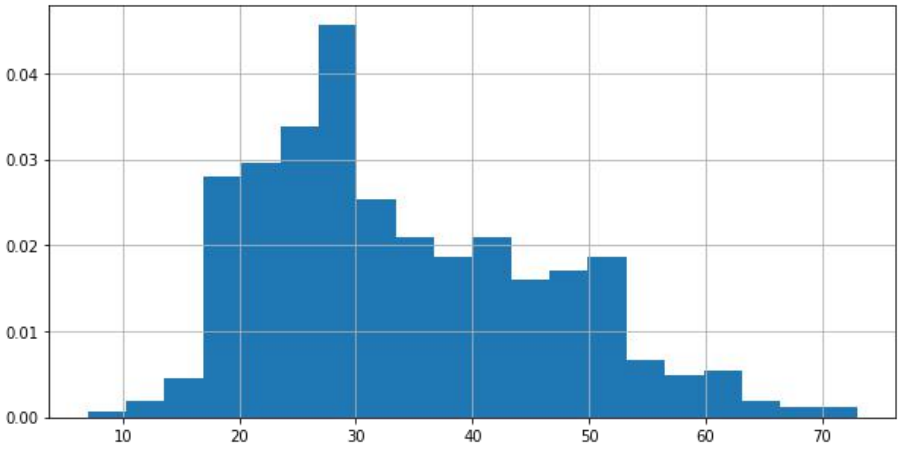

图 2.11 年龄分布

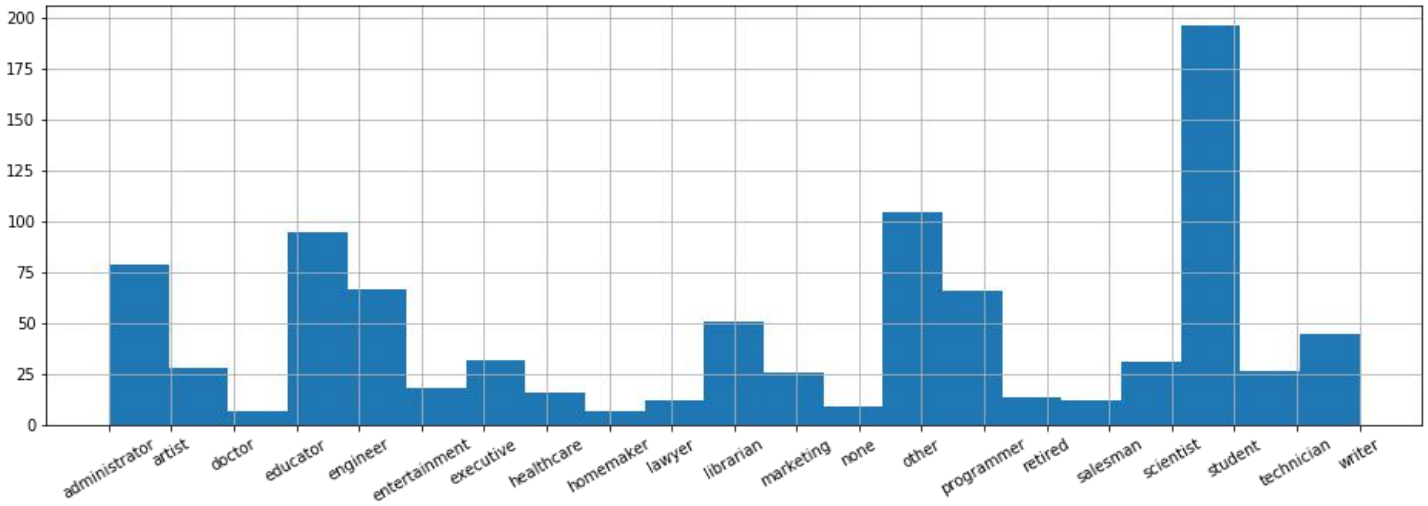

图 2.12 职业分布

（2）电影数据

电影数据由电影（MovieID）、电影名（Title）和电影风格（Genres）三个字段组成。电影风格有 18 种，分别为“动作”、“冒险”、“卡通”、“儿童”、“喜剧”、“犯罪”、“记录片”、“戏剧”、“幻想”、“黑色电影”、“恐怖”、“音乐”、“神秘”、“浪漫”、“科幻”、“惊悚”、“战争”和“西方” 。

如图 2.12 所示，每部电影可能有多种风格，原始数据由“|”分隔。数据集中主要是喜剧和戏剧，具体类型分布如图 2.13 所示。

	movie_id	title	genres
0	1	Toy Story (1995)	Animation|Children's|Comedy
1	2	Jumanji (1995)	Adventure|Children's|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama
4	5	Father of the Bride Part II (1995)	Comedy

图 2.13 电影数据前五行

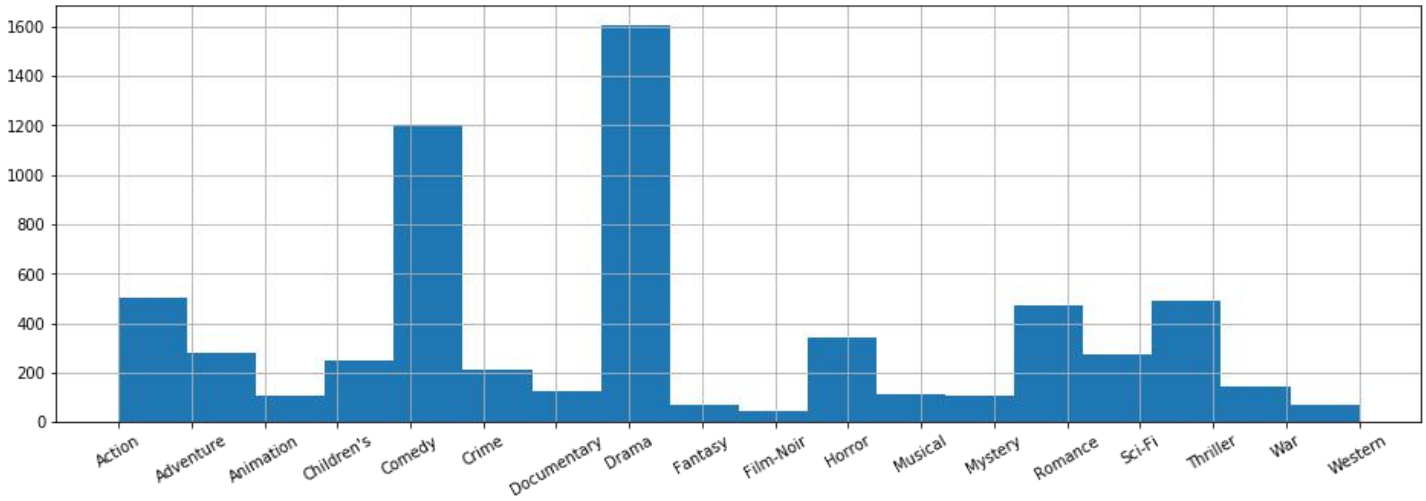

图 2.14 电影类型分布

（3）评分数据

评分数据由用户（UserID）、电影（MovieID）、评分（Rating）和时间戳（Timestamp）四个字段组成。评分值是 1 到 5 的整数。评分数据极度稀疏，评分矩阵中评分比例占比为 4.26%，符合协同过滤数据的特性。

2.4.2 评价指标

推荐系统有许多目标，根据不同目标，有不同的评价指标。这些评价指标可以用于定性或定量地评价推荐系统各方面的性能。

（1）准确度

预测准确度衡量推荐算法对推荐任务的推荐准确性，是最重要的一个评测指标。指标高低反映了算法对用户喜好的建模能力。

1）评分预测准确度

对于提供打分功能的网站，例如豆瓣，预测用户对项目的评分，称为评分预

测。一般使用 RMSE 或者 MAE 计算准确度。假设 r_{ui} 是真实评分数据，$\hat{r}_{ui}$ 是算法的预测评分。

RMSE 定义为：

$$\mathrm{RMSE}=\sqrt{\frac{\sum_{u,i\in T}(r_{ui}-\hat{r}_{ui})^2}{|T|}} \tag{2.38}$$

MAE 定义为：

$$\mathrm{MAE}=\frac{\sum_{u,i\in T}|r_{ui}-\hat{r}_{ui}|}{|T|} \tag{2.39}$$

Netflix 认为“RMSE 增大了对预测不准用户的评分惩罚，因此对系统的评测更加苛刻，如果评分被限制为整数的话，那么对预测结果取整会降低 MAE 的误差”[10]。

2）TopN 推荐准确度

对于电商网站，例如淘宝，会为不同的用户推荐若干个物品。为用户推荐 N 个物品的行为，称为 TopN 推荐。假设 R(u)是算法给出的 N 个推荐物品，T(u)是测试集中用户的已交互物品的集合。

准确率定义为：

$$\mathrm{Precision}=\frac{\sum_{u\in U}|R(u)\cap T(u)|}{\sum_{u\in U}|R(u)|} \tag{2.40}$$

召回率定义为：

$$\mathrm{Recall}=\frac{\sum_{u\in U}|R(u)\cap T(u)|}{\sum_{u\in U}|T(u)|} \tag{2.41}$$

（2）覆盖率

覆盖率（coverage）度量推荐算法发掘长尾物品的能力，是个性化推荐很重要的指标之一。覆盖率有很多种定义方法，假设用户集合 U，推荐列表为 R(u)，则最简单的覆盖率定义为：

$$\mathrm{Coverage}=\frac{|\mathrm{U}_{u\in U}R(u)|}{|I|} \tag{2.42}$$

这样简单地定义覆盖率，无法很好地度量长尾发掘能力，如果推荐算法将所

有物品均匀推荐给每个用户，那么按照公式 2.42 计算，能够得到很高的覆盖率，但实际上对长尾的发掘能力为零。

1）项目覆盖率

为了度量推荐算法对项目的覆盖率，对公式 2.42 改进，判断的是推荐算法成功推荐的项目，而不是推荐项目的并集。假设用户 u 在测试集上的行为列表为 T(u)，用户集合为 U，项目集合为 I，项目覆盖率为：

$$\text{ItemCoverage} = \frac{|\mathrm{U}_{u \in U}(R(u) \cap T(u))|}{|I|} \tag{2.43}$$

2）用户覆盖率

不同的用户有不同喜好的长尾物品。项目覆盖率高的推荐算法可能只对部分用户发挥了较强的长尾发掘能力，而用户覆盖率度量了推荐算法长尾发掘能力的普适性。I(・)是示性函数，I(true)=1，I(false)=0，例如 I(1>0)=1。

$$\text{UserCoverage} = \frac{\sum_{u \in U} I(|R(u) \cap T(u)| > 0)}{|\mathrm{U}|} \tag{2.44}$$

（3）短时预测成功率

短时预测成功率（short item prediction success，sps.），度量的是推荐系统对下一次用户感兴趣/会交互的那一个物品的预测成功率。sps 对时间很敏感，在 TopN 任务上，比准确率和召回率更严格。sps 要求推荐算法对于一个用户，给出 N 个推荐物品，用户的下一个希望交互的物品在推荐列表内则推荐成功。sps 适合用于评测会话型的推荐场景。

2.4.3 数据处理

对于标准协同过滤问题，只考虑用户和物品的交互数据，所以实验中所有方法都只使用 MovieLens 1M 的评分数据。

每个用户的评分序列按时间升序排序，形成时间序列，前半部分当作观测数据，后半部分当作期望推荐数据。

数据以用户为单位进行划分，随机 600 名用户（及其评分）分入交叉验证集合，随机 600 名用户分入测试集，剩下分入训练集。

实验任务是 TopN 推荐，默认设置 N=10。

对于在测试集中的每名用户，算法根据训练集建立模型，根据该用户的观测

数据给出推荐。算法使用的评测指标为短时预测成功率、召回率、项目覆盖率和用户覆盖率，具体描述见 2.4.2 节。除了基于矩阵分解的算法使用到了评分值，其它的算法都将 1 到 5 的评分显性反馈数据当作 0 或 1 的隐形反馈数据使用。

2.4.4 隐层参数维度的影响

在 2.2 节中提到，在 RNN 的重复模块中，隐层参数维度是一个非常重要的参数。隐层参数维度又称单元大小（cell size）或者神经元个数（neurons）。如果维度过小，那么 RNN 模型就没有办法学习到足够的信息，但是如果维度过大，就会导致训练时间过长并且增加过拟合的风险。

实验对比了 LSTM 和 GRU 两种 RNN 模型。固定训练轮次，当 LSTM 和 GRU 的维度相同时，两个方法能力相近，但 GRU 耗时比 LSTM 短。

实验固定遍历数据集 48 轮，迭代 236384 次，对比了 32 维的 LSTM 和 8、16、32、64 维的 GRU，训练总耗时如表 2.1 所示：

表 2.1　耗时对比

方法	总耗时（秒）
GRU-8	4238
GRU-16	4241
GRU-32	4293
GRU-64	4762
LSTM-32	5685

在短时预测成功率、召回率、项目覆盖率和用户覆盖率四个指标下，LSTM-32、GRU-8、GRU-16、GRU-32 和 GRU-64 随遍历训练训练集次数增加，结果的对比图如下所示。32 维的 LSTM 与 32 维的 GRU 在四个指标上表现相近，但是 LSTM 的总耗时大于 GRU，实验结果表明，基于 GRU 的协同过滤算法好于 LSTM。8 维的 GRU 在短时预测成功率、召回率、项目覆盖上略逊于 16、32、64 维的 GRU，随着维度的提高，指标会略微提升，但是维度越大耗时越长。在精度和耗时之前的权衡下，实验默认采用 32 维的 GRU。

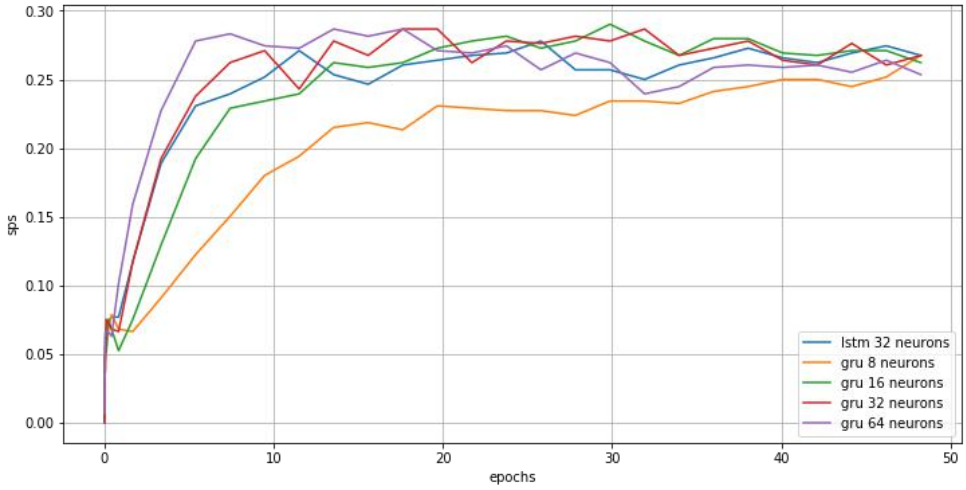

图 2.15　轮次-短时预测成功率

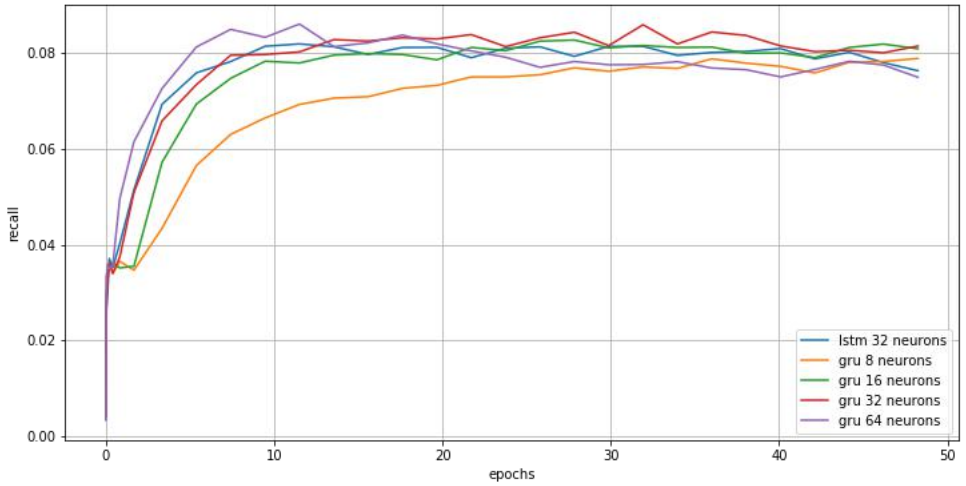

图 2.16　轮次-召回率

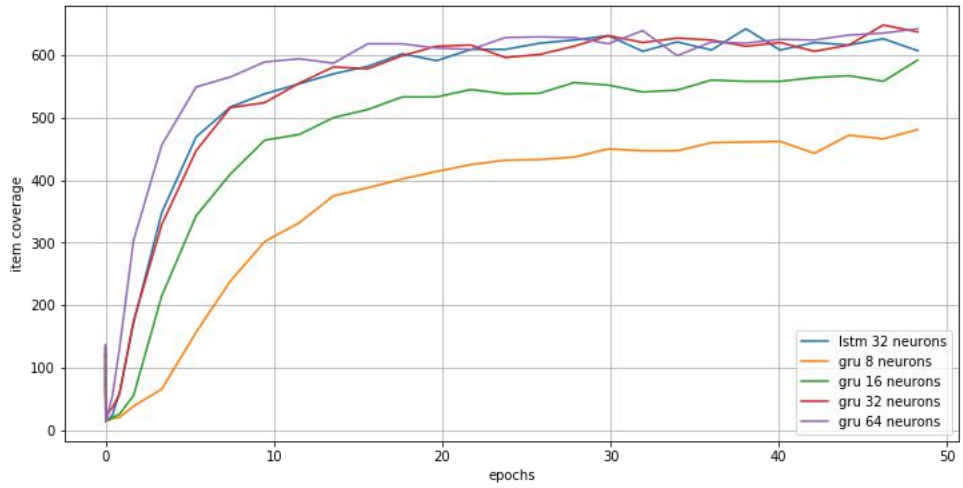

图 2.17　轮次-项目覆盖数

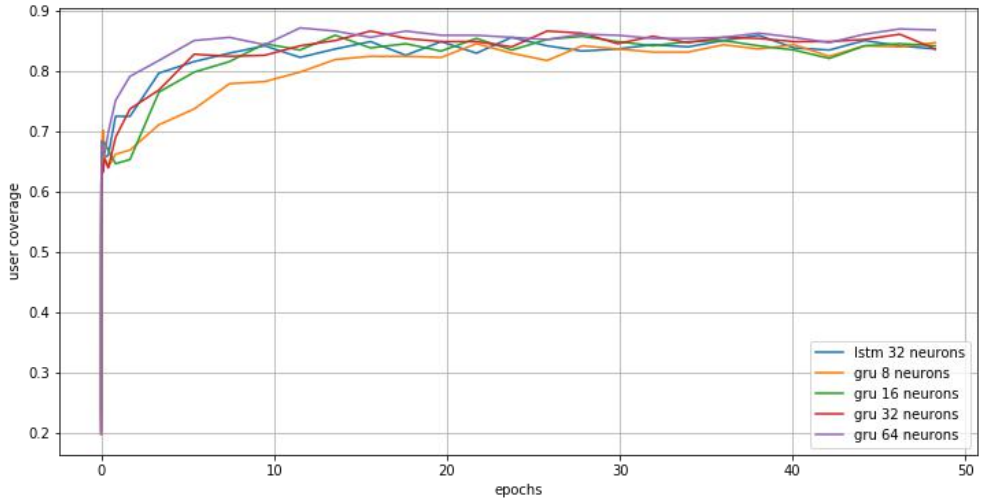

图 2.18　轮次-用户覆盖率

2.4.5 迭代优化算法的影响

大部分机器学习算法，都分为参数梯度计算和参数梯度更新两个步骤。GRU 的参数梯度计算使用反向传播算法，推导公式见 2.2.2 节。参数梯度更新涉及到优化理论，常见的有利用一阶导性质的批量梯度下降法、随机梯度下降法等，利用二阶导性质的牛顿法、L-BFGS 等。

GRU 属于神经网络的一种，而训练神经网络一般使用的是一阶导优化算法。常见的有随机梯度下降、动量法、Nesterov、Adam 等。

动量法通过引入物理中的动量概念，来加速训练。假设 J(θ)是损失函数，Δ_{θ}J(θ)是损失函数关于 θ 的梯度， α 是学习率。动量更新公式如下：

$$V(t)=\lambda V(t-1)+\alpha\nabla_{\theta}J(\theta) \tag{2.45}$$

其中 λ 是动量超参数，一般设置为 0.9。最后使用动量更新参数，θ = θ -V(t)。

Nesterov 梯度加速法在动量法的基础上，改进了公式，改进的动量更新公式如下：

$$V(t)=\lambda V(t-1)+\alpha\nabla_{\theta}J(\theta - \lambda V(t-1)) \tag{2.46}$$

最后同样也是使用动量更新参数， θ = θ -V(t)。

Adam（Adaptive Moment Estimation），是一种学习率自适应优化算法，能够解决学习率消失等问题[11]。虽然计算过程更加的复杂，但是收敛速度快，是训练深度学习模型最常用的方法。

实验对比了 Nesterov、Adam 两种优化算法在隐藏参数维度 32 的 GRU 的优化效果。实验固定训练时间为 3600 秒，最终效果如下图所示：

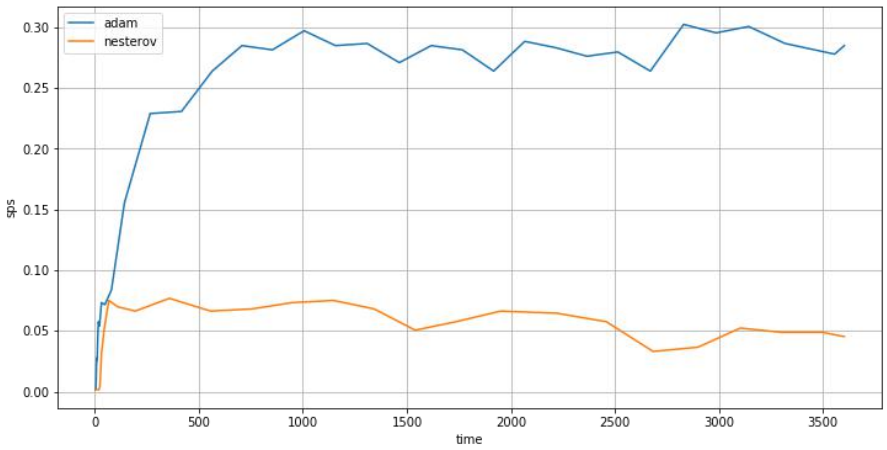

图 2.19　时间-短时成功率

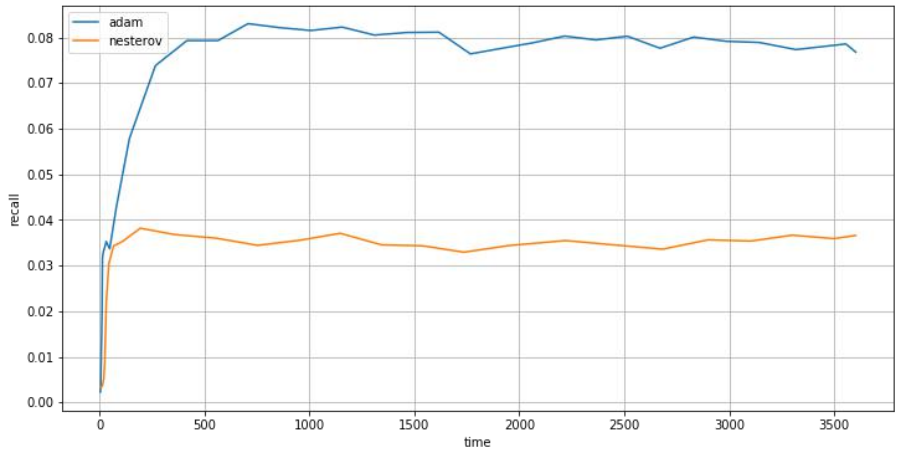

图 2.20　时间-召回率

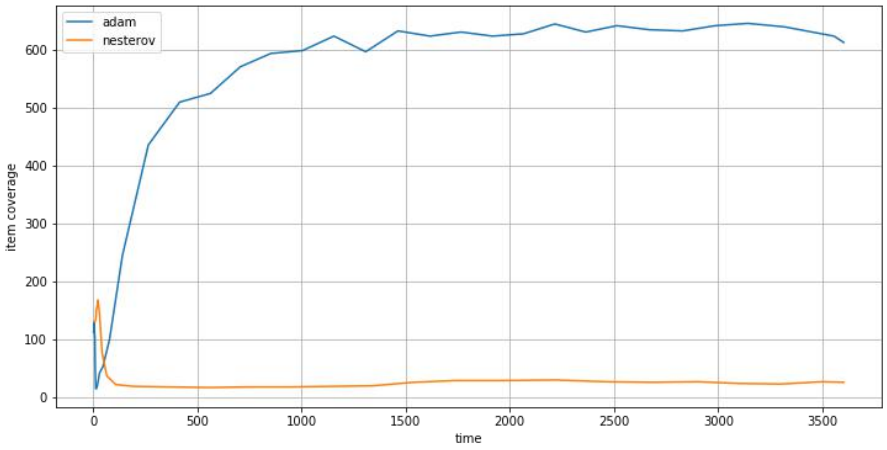

图 2.21　时间-项目覆盖数

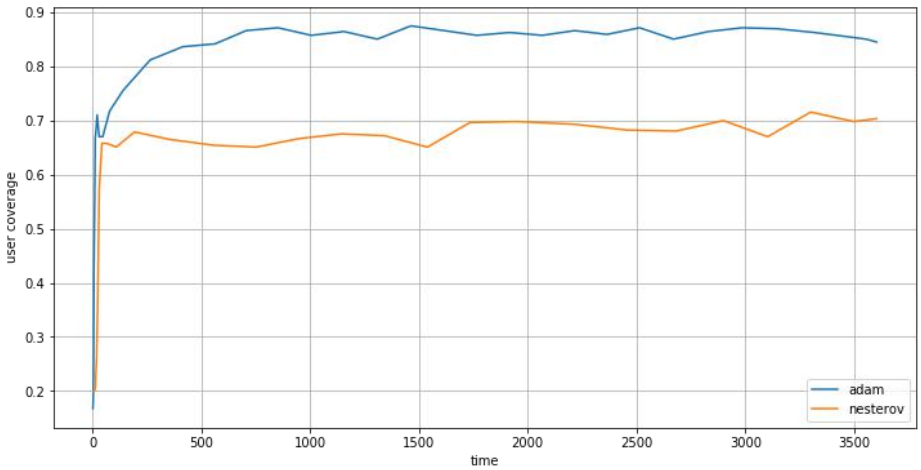

图 2.22　时间-用户覆盖率

从实验中可以得出结论，在训练 GRU 模型时，adam 优化算法远比 Nesterov 好。对于深度学习模型，目标函数是非凸的，并且有大量局部最优解，优化算法的目标之一就是摆脱糟糕的局部最优解。在所有指标上，adam 优于 Nesterov，特别在项目覆盖上，可以看到 Nesterov 陷入的局部最优区域远差于 adam。

2.4.6 Dropout 的影响

Dropout 方法就是在训练过程中，对某层的单元的输出按一定概率置零。在测试阶段和部署阶段，屏蔽 Dropout。Dropout 相当于对神经网络做了正则化，在提高神经网络的效果同时，一定程度上缓解了过拟合现象。

Dropout 相当神经网络的特殊正则化。在每轮迭代中，通过 Dropout 层的输入会以预设概率置零输出，因此网络不知道 Dropout 会让哪些单元通过。比起之前能够使用所有单元，网络可以专注于某些单元来达到较好的效果，此时这些单元可能会被 Dropout 屏蔽，从而迫使模型分散注意力，依赖所有单元来做预测。因为网络的注意力分散，使得所有单元特征都能得到关注，减低了高权重的单元特征，相当于为网络做了正则化。

同时，Dropout 隐含了集成学习的能力。在训练网络模型时，Dropout 会随机置零一部分通过它的单元。因此，相比于直接训练一个网络模型，添加 Dropout 层后，每一轮都在前一轮训练出来的模型基础上，使用部分网络（未被 Dropout 屏蔽部分）训练一个新的小模型，最终的网络模型是之前小模型的组合。

在 RNN 中，一般只在不同层循环模块之间使用 Dropout，而不在同一层的循环模块之间使用。也就是说从时刻 t-1 传递到时刻 t 时，RNN 神经网络不会进

行状态的 Dropout，而在同一时刻 t 中，不同循环模块之间会使用 Dropout。如图 2.22，实线箭头表示不使用 Dropout，虚线箭头表示使用 Dropout。

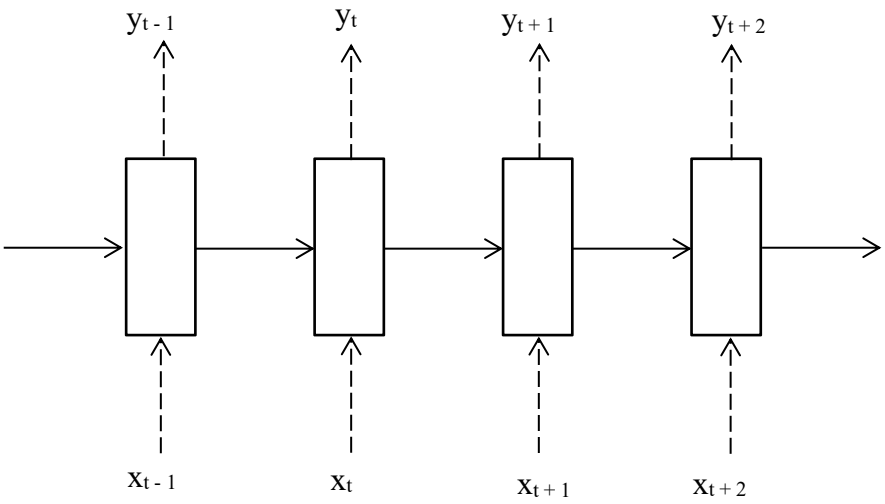

图 2.23　RNN 使用 Dropout 示意图

实验采用 GRU-32 模型，adam 优化算法。遍历训练集 48 轮，对比 Dropout 从 0 到 0.9 的屏蔽概率，在短时预测概率和召回率上的影响，结果如图 2.23 所示：

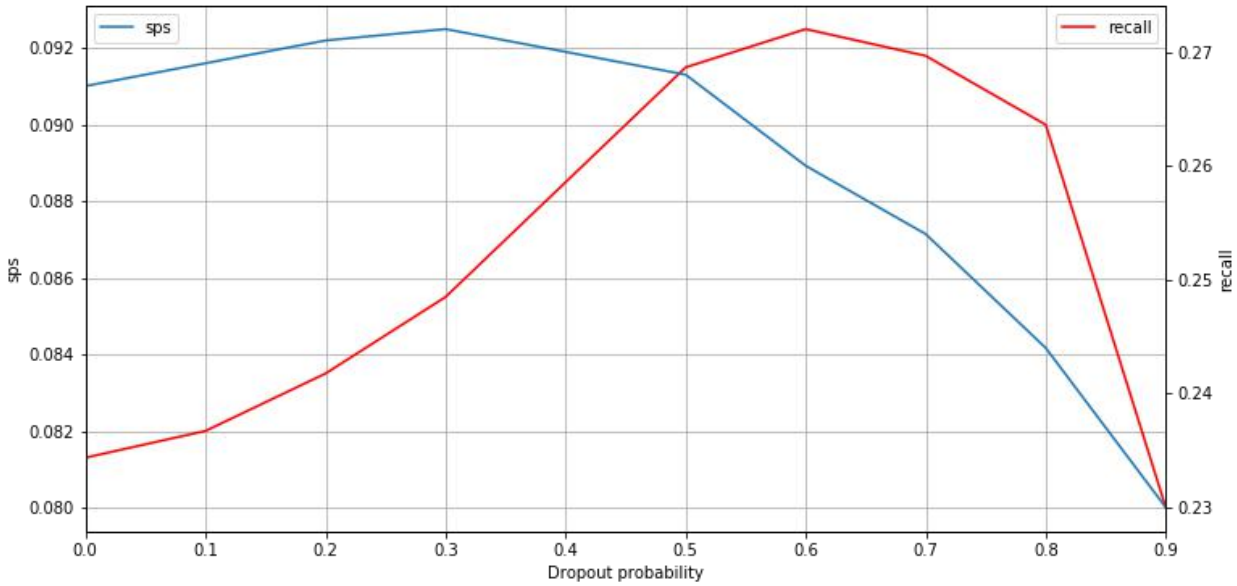

图 2.24　Dropout 的影响

由实验结果可知，当 Dropout 概率上升时，recall 不断上升，在 0.6 左右开始下降，而 sps 在短暂上升后，0.3 左右开始下降。对于小概率的 Dropout，sps 和 recall 的精度同时提高，可见适当地 Dropout，会提升模型整体的能力。

2.4.7 TopN 的影响

TopN 推荐任务，旨在为每名用户推荐 N 个物品，以满足个性化推荐需求。这里 N 是一个超参数，需要预先设定。在 2.4.3 节有说明，实验默认设置 N=10，本小节，将研究参数 N 对各个指标的影响。

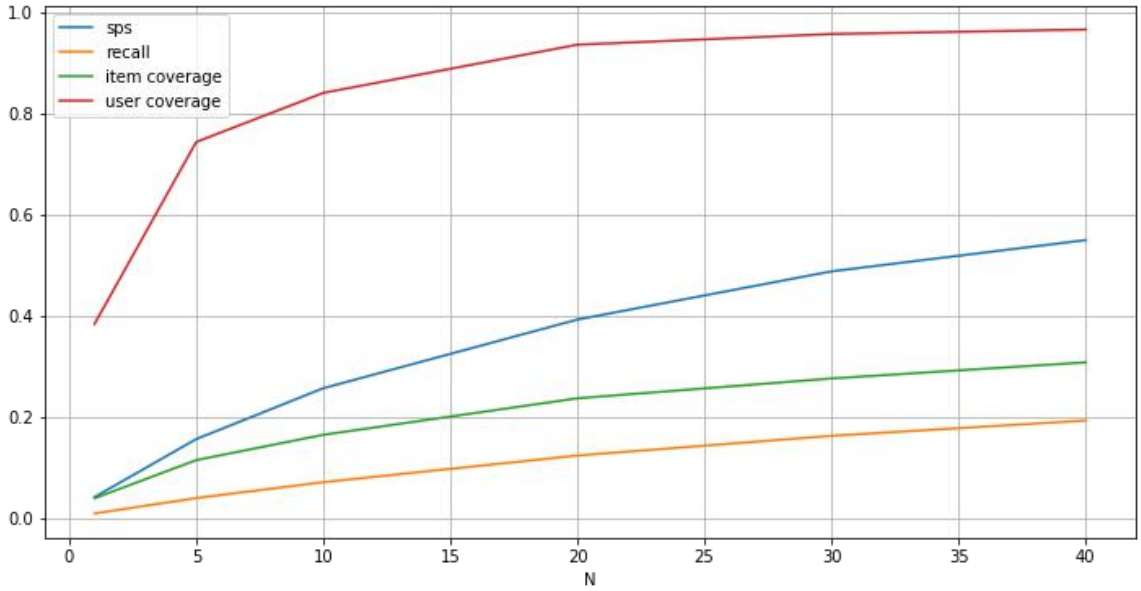

图 2.25　TopN 的影响

之前实验为了更明显地对比，项目覆盖以个数为单位，这里为了和其它三个指标绘制在同一张图，与用户覆盖一样进行了归一化，除以有效电影数 3706。

实验表明，随着 N 的增大，所有指标都在上升。如果 N 等于所有物品数，则所有指标为最大值，但是这没有任何实际意义。实际应用环境中，在某个场景下，只有可能推荐给用户有限个物品，且一次性推荐的物品越多，个性化推荐的意义越低。需要在参数 N 的增大对于指标提升和对推荐个性化的削弱中做出权衡。对于 N 的设置，需要进行在线实验，通常使用的方法是 A/B 测试。在本实验，除了本小结以外，都默认设置 N=10。

2.4.8 各种方法实验结果对比

本小节，将基于 GRU 的协同过滤算法与传统协同过滤算法对比。基于用户的协同过滤算法，是使用最广泛的推荐算法之一，原理见 1.3.1 节，它有一个参数 k，代表寻找与目标用户最近似的 k 个用户，利用他们的喜好来为目标用户推荐物品，k 的影响见表 2.2。参数 k 对 sps 和 recall 影响不大，但随着 k 的增大项目覆盖数降低，用户覆盖数增大。

表 2.2　基于用户的参数 k 的影响

	10	20	30	40	50	60	70	80	90	100
sps	0.104	0.120	0.118	0.127	0.113	0.118	0.127	0.116	0.121	0.118
recall	0.050	0.055	0.056	0.058	0.057	0.057	0.059	0.058	0.058	0.057
item 覆盖	423	383	357	348	322	307	310	301	289	282
user 覆盖	0.725	0.746	0.750	0.769	0.773	0.768	0.780	0.781	0.783	0.785

实验还对比了马尔科夫模型，它很适合用于建模时序数据。此时，物品即状态，问题变成根据当前的状态序列，预测之后状态序列。实验采用的是一阶马尔可夫模型，即只考虑用户的最后一次评价。对于基于矩阵分解的方法，对比实验中采用的是 Funk-SVD，详见 1.3.2 节。最终结果如表 2.3 所示：

表 2.3　各种方法对比结果

方法	sps(%)	recall(%)	item 覆盖	user 覆盖(%)
基于用户 (k=80)	11.62	5.77	301	77.99
基于矩阵分解 (Funk-SVD)	11.28	5.65	399	80.8
时序化+马尔科夫链	19.54	5.27	558	78.70
时序化+GRU (32 维,adam, Dropout-50%)	29.47	7.63	637	86.98

2.4.9　实验结果总结

实验结果表明，基于 GRU 的方法在精度上远超过其它方法，且用户覆盖率和项目覆盖率同时高于其它方法，说明其在发掘长尾物品上有很大优势。

第 3 章 基于主动学习的改进

本章提出基于改进的 MinRating 的主动学习算法，加速 GRU 的训练。最终提出基于 GRU 的主动学习协同过滤算法。

3.1 主动学习

主动学习的假设是，机器学习算法要根据已学到的信息主动去选择数据，使用这些数据来训练模型，然后再去选择数据，不断循环，如图 3.1 所示。当样本只有少量有被标记时，使用主动学习，让模型主动选择最有效的几个数据，然后让专家来打标签，再喂给模型训练，可以有效地减少标记样本的代价。当样本有足够的标记时，使用主动学习，能够在使用更少的样本、更少的训练轮次下，获得更好的模型。

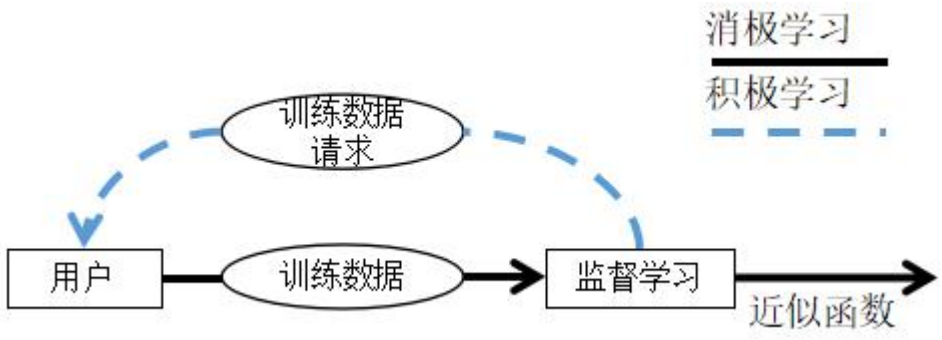

图 3.1　主动学习示意图

主动学习的本质是动态采样数据，主要的方法如表 3.1 所示：

表 3.1　基于用户的参数 k 的影响

方法	描述/目标	可能的限制条件
基于不确定性	减少以下内容的不确定性 ● 评分评估 ● 决策边界 ● 模型参数	降低不确定性可能并不能总是提高准确率；该模型可能只对错误的事情肯定（如用错了预测方法）
基于误差	通过错误与以下内容的关系来减少预测错误 ● 输出评估中的变化 ● 测试集错误 ● 参数评估中的变化 ● 参数估计值的方差	可靠地评估错误的减少是困难并且计算昂贵的

方法	描述/目标	可能的限制条件
基于组合	基于以下内容之间的共识找出有用的训练集 ● 聚类中的模型 ● 多重候选模型	由于其效用取决于模型/候选的质量且可以是计算昂贵的，因为它的执行与多重模型/候选有关

假设用户 u 对电影 m 的评分值为 $r_{u,m}$ ，模型给出的预测数值为 $\hat{r}_{u,m}$，一种看法是$|\hat{r}_{u,m}-r_{u,m}|$最大的那部电影，才是信息量最大的电影，即对模型训练影响最大的电影。但是事实上，无法获知用户 u 对所有电影的评分，所以无法使用这个公式判断电影的信息量。因为用户大多会去评价或者交互自己感兴趣的电影/物品，所以预测的误差更多的出现在对用户评高评分的电影预测低分的情况，而动态采样模型评分预测中，评分值最低的项目数据，这种方法称作 MinRating[12]。

因为探讨的是 TopN 预测任务，MinRating 算法无法使用。GRU 输出的是所有项目的概率，最终会选取概率最高的 N 个项目进行推荐。这里对 MinRating 做改进，以适应 TopN 推荐。每次选取 GRU 的 Top1 推荐（即，给每名用户推荐项目中置信度最高的项目）中置信度最低的用户。假设 GRU 给出用户 u 对物品 i 的喜好预测值为 O_{ui} ，则主动学习算法主动挑选的用户为：

$$u^{*}=\arg\min_{u}(\max_{i}(O_{ui})) \tag{3.1}$$

挑选的用户及其评分序列加入训练集，待下一轮训练使用。

3.2 基于 GRU 的主动学习协同过滤算法

基于 GRU 的主动学习推荐算法，输入是序列化的协同过滤数据，详见 2.1 节，主体模型是 GRU，输出的是所有项目的概率，通过排序得到最大的 N 个项目，将 N 个项目组成推荐列表完成 TopN 推荐任务，训练过程使用 3.1 节描述的主动学习算法，使得 GRU 能在少量轮次下获得更优效果。

算法具体流程如图 3.2 所示：

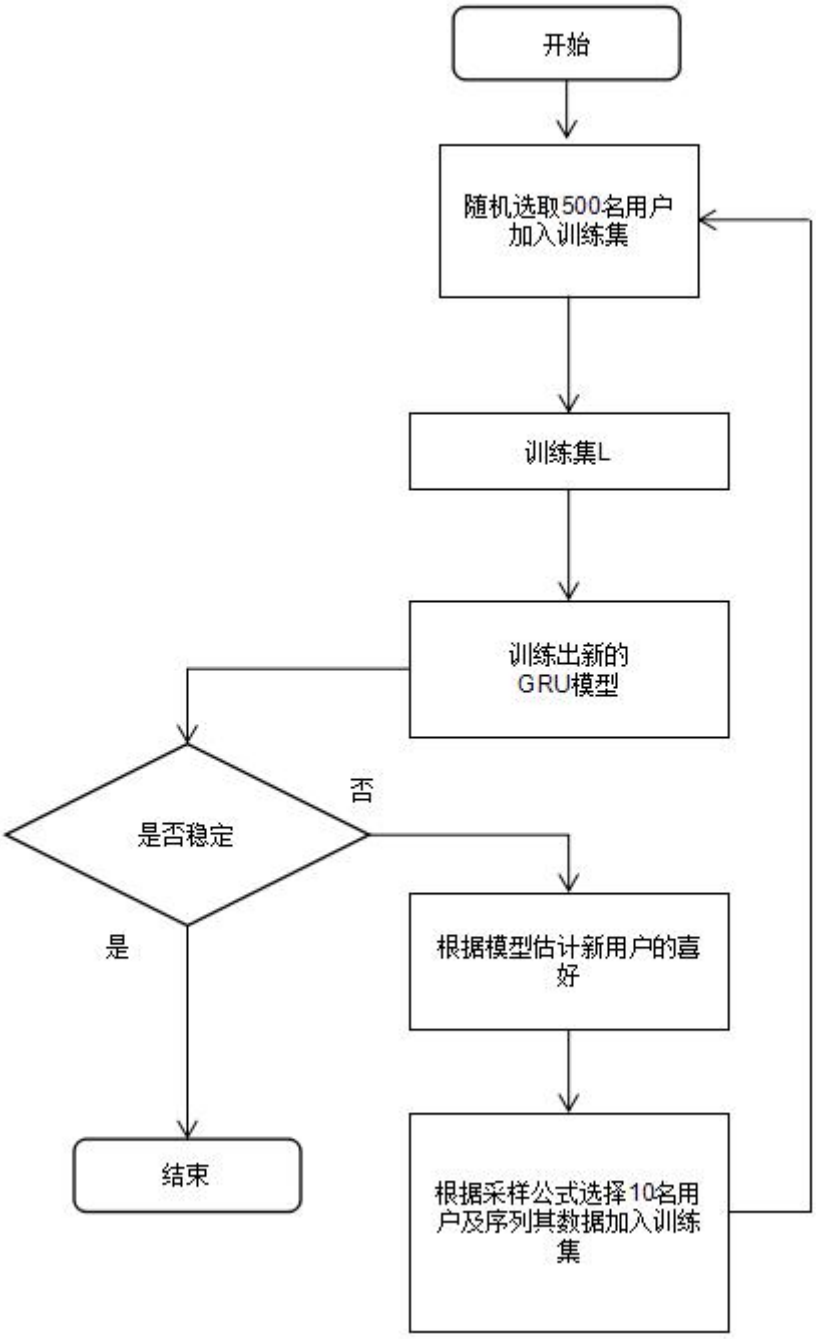

图 3.2　算法流程图

3.3　实验验证

3.3.1 实验结果

实验使用隐层参数维度 32 的 GRU 进行对比实验，使用 adam 优化算法，未添加 Dropout 层，固定遍历 50 轮训练集，分为使用主动学习和不使用主动学习两组实验。

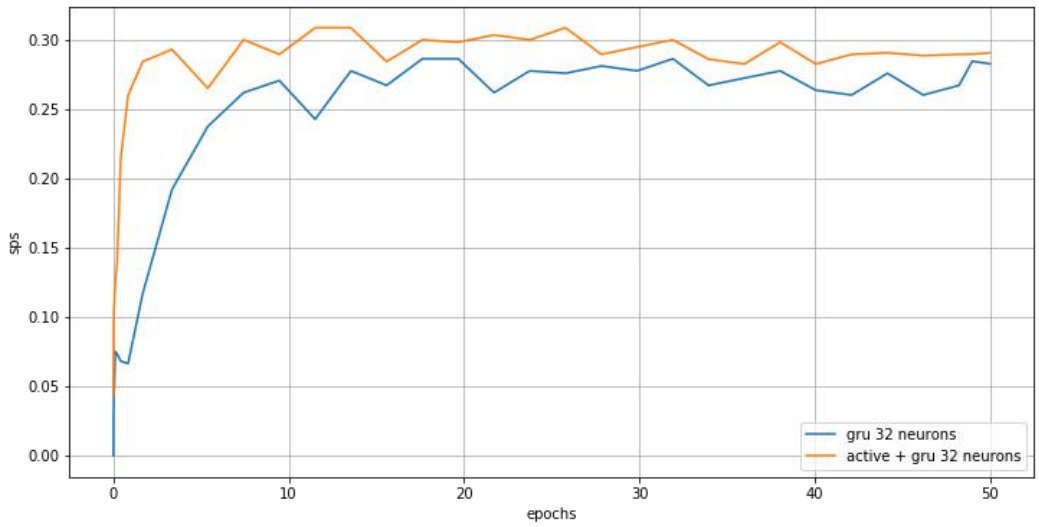

图 3.3　轮次-短时预测成功率

3.3.2　实验结果分析

实验结果表明，使用主动学习，能以更少的遍历轮次获得更高的效果。同时，使用主动学习，降低 GRU 的样本需求度，能够在一定程度上解决冷启动问题。

第 4 章 总结与展望

本论文提出基于 GRU 的主动学习协同过滤推荐算法，结合时序化、GRU 和主动学习解决协同过滤问题。利用时间维度信息，将协同过滤数据时序化；使用 GRU 对时序数据建模，将 N 个最大概率的输出作为 TopN 的推荐结果；使用主动学习，动态采样数据，加速模型训练。

实验表明，算法在短时预测成功率、召回率、项目覆盖和用户覆盖上优于传统协同过滤算法。该算法能够有效提升 TopN 任务的精度，并且因为同时提升了项目覆盖和用户覆盖，具有较强的发掘长尾物品的能力。算法中的主动学习部分能够帮助模型快速建立，可以在一定程度上解决系统的冷启动问题。

算法还存在进一步改进的空间：

（1）只尝试了一种主动学习算法。可以考虑选择更多主流的主动学习算法，进行对比。

（2）算法流程中，每轮挑选的用户数，这是个超参数，需要手动调节，不能自适应动态调整。

参 考 文 献

[1] 冷亚军,陆青,梁昌勇.协同过滤推荐技术综述[J]. 模式识别与人工智能,2014,27(08):720-734.

[2] Billsus D, Pazzani M J. Learning Collaborative Information Filters[C]// International Conf on Machine Learning. 1998.

[3] Funk S. Netflix Update: Try This at Home[EB/OL]. 引自个人博客, http://sifter.org/~simon/journal/20061211.html, 2006.

[4] Koren Y. Factor in the neighbors:Scalable and accurate collaborative filtering[J]. Acm Transactions on Knowledge Discovery from Data, 2010, 4(1):1-24.

[5] He X, Liao L, Zhang H, et al. Neural Collaborative Filtering[C]// The International Conference. 2017:173-182.

[6] Xue H J, Dai X Y, Zhang J, et al. Deep matrix factorization models for recommender systems[C]// International Joint Conference on Artificial Intelligence. AAAI Press, 2017:3203-3209.

[7] Sathasivam S, Abdullah W A T W. Logic Learning in Hopfield Networks[J]. Modern Applied Science, 2008, 2(3):57.

[8] Hochreiter S, Surhone J. Long short-term memory[J]. Neural Computation. 1997. 9(8):1735~1780.

[9] Cho K, Van Merrienboer B, Gulcehre C, et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation[C]. Empirical Methods in Natural Language Processing, 2014:1724-1734.

[10] Tikk D. Major components of the gravity recommendation system[J]. Acm Sigkdd Explorations Newsletter, 2007, 9(2):80-83.

[11] Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.

[12] Karimi R, Freudenthaler C, Nanopoulos A, et al. Non-myopic active learning for recommender systems based on Matrix Factorization[C]// IEEE International Conference on Information Reuse and Integration. IEEE, 2011:299-303.

致　谢

本论文在王成老师的指导下完成，在此，向老师表示由衷的谢意。老师每周都会召开会议亲自给我指导，在研究方向和思路上给予了很大的帮助。老师严谨治学的理念深深影响了我，在未来的学习生活中我会以更加深刻的角度看待问题。

同时，我也要感谢我的家人、朋友们，在我学习和生活中陪伴着我。感谢你们一路的陪伴，我会更加坚定地走下去，不负期望。

作者名　潘傲寒

2018　年 5 月 24 日