融合动态协同过滤和深度学习的推荐算法

邓存彬[1,2] 虞慧群[1] 范贵生[1]

(华东理工大学计算机科学与工程系 上海 200237)[1] (上海市计算机软件测评重点实验室 上海 201112)[2]

摘 要 在信息爆炸的时代,推荐系统在减轻信息过载方面发挥了巨大的作用。目前,推荐系统普遍使用传统的协同过滤算法学习用户商品行为矩阵中的隐向量,但是其存在数据稀疏性和冷启动问题,同时未考虑用户的兴趣偏好以及商品的受欢迎程度会随时间发生改变,这极大地限制了推荐的准确性。已有学者利用深度学习模型学习辅助信息的特征来扩充协同过滤算法的特征,取得了一定的成果,但并未充分有效地解决全部问题。以电影推荐为研究对象,提出了融合动态协同过滤和深度学习的推荐算法。首先,利用动态协同过滤算法融入时间特征;然后,利用深度学习模型来学习用户和电影特征信息,以形成高维潜在空间的用户特征和电影特征的隐向量;最后,将其融入到动态协同过滤算法中。以MovieLens为实验数据集对电影的评分进行预测,实验结果表明所提算法提高了电影评分预测的准确性。

关键词 电影推荐,隐向量,深度学习,动态协同过滤

中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.08.005

Integrating Dynamic Collaborative Filtering and Deep Learning for Recommendation

DENG Cun-bin[1,2] YU Hui-qun[1] FAN Gui-sheng[1]

(Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China)[1]

(Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai 201112, China)[2]

Abstract In the era of information explosion, the recommendation system plays an enormous role in reducing information overload. At present, the recommendation system generally uses the traditional collaborative filtering algorithm to learn the hidden vector in the user-item behavior matrix, but it has the problem of data sparseness and cold start, and does not consider the customer preferences and the popularity dynamics of items. This greatly limits the accuracy of the recommendation system. Some scholars have used the deep learning model to learn the features of the auxiliary information to enrich the features of the collaborative filtering algorithm, and achieved certain results, which does not fully solve all the problems. This paper took film recommendation as the research object, and proposed a recommendation algorithm that combines dynamic collaborative filtering and deep learning. Firstly, the dynamic collaborative filtering algorithm incorporates temporal features. Secondly, it uses deep learning model to learn user and movie feature information to form the hidden vector of user features and movie features in high-dimensional latent space. Finally, it is integrated into the dynamic collaborative filtering algorithm. Extensive experiments on MovieLens datasets show that the proposed method improves the accuracy of film score prediction.

Keywords Movie recommendation, Hidden vector, Deep learning, Dynamic collaborative filtering

1 引言

随着大数据、云计算和物联网等技术的迅速发展,数据规模呈爆炸式的增长,在满足用户对信息需求的同时,也带来了严重的信息过载问题,推荐系统应运而生。目前推荐系统已被应用于多个领域,在视频推荐领域,Netflix比赛是标志性的视频推荐系统比赛,旨在提高视频推荐系统的推荐效果;另外,在电子商务平台领域中,例如eBay、亚马逊、天猫和京东等,推荐系统的推荐效果对电子商务的发展有着巨大的影响[1]。个性化推荐系统是推荐系统最重要的研究方向,其根据用户需求、兴趣等,利用推荐算法从海量数据中挖掘出用户感兴趣的项目(如信息、服务、物品等),并将结果以个性化列表的方式推荐给用户。个性化推荐系统中最关键的部分就是推荐算法,它的好坏在一定程度上直接决定了推荐系统的性能。

到稿日期:2018-07-08 返修日期:2018-11-02 本文受国家自然科学基金(61702334,61772200),上海市浦江人才资助计划(17PJ1401900),上海市自然科学基金资助项目(17ZR1406900,17ZR1429700),华东理工大学教育科研基金(ZH1726108),上海应用技术学院资助合作创新基金会(XTCX2016-20)资助。

邓存彬(1993—),男,硕士生,主要研究方向为数据挖掘、机器学习;**虞慧群**(1967—),男,教授,博士生导师,CCF高级会员,主要研究方向为软件工程、形式化方法,E-mail:yhq@ecust.edu.cn(通信作者);**范贵生**(1980—),男,副研究员,CCF会员,主要研究方向为软件工程、可信计算。

在大量的个性化推荐算法中，传统的协同过滤算法是被应用得最普遍的一种算法。基于模型的协同过滤算法（矩阵分解（MF）的协同过滤）是将用户对项目的行为通过矩阵分解成用户隐向量和项目隐向量，再通过隐向量的内积进行推荐。协同过滤算法依赖于用户对项目的行为数据，而不需要推荐对象的其他信息，对所有的推荐项目都适用，可以用来推荐电影、音乐、新闻等项目。但是，协同过滤算法存在数据稀疏性和冷启动问题，同时未考虑用户的兴趣偏好以及商品的受欢迎程度会随时间发生改变，这极大地影响了推荐的准确性。

近年来，深度学习在图像处理、自然语言处理和语音识别等领域取得了突破性进展。深度学习可以通过学习一种深层次非线性网络结构，来表征用户和项目相关的海量数据，具有强大的从样本中学习数据集本质特征的能力，能够获取用户和项目的深层次特征表示。此外，深度学习还可以从多源异构数据中进行自动特征学习，将不同的数据映射到一个相同的隐空间，从而获得数据的统一表示。

本文以电影推荐中的电影评分预测为研究对象，在 MF 的协同过滤算法的基础上，考虑了用户的兴趣偏好以及电影的受欢迎程度会随时间发生变化，将时间因素融入到 MF 的协同过滤算法中，以形成动态协同过滤算法。利用深度学习模型来学习用户特征和电影特征，以形成高维潜在空间内的用户特征隐向量和电影特征隐向量，并将其融入到动态协同过滤算法中，以解决动态协同过滤算法中用户电影评分矩阵的稀疏性和冷启动问题，提高了电影评分预测的准确性。

本文提出的方法与其他方法相比，存在以下特征：1）充分考虑用户特征和电影特征，并利用深度学习模型中的卷积神经网络（Convolutional Neural Networks，CNN）和多层感知机（Multi-Layer Perception，MLP）来获取深层次的表示特征；2）融合动态协同过滤算法和深度学习模型来构成混合推荐算法，并对动态协同过滤算法的评分预测函数和损失函数进行了改进，形成了混合推荐算法的评分预测函数和损失函数。

2　相关工作

推荐系统是一个活跃的研究方向，许多学者已经对此进行了大量的研究。本节将介绍与本文相关的推荐算法的研究现状，包括协同过滤算法、深度学习模型及混合推荐算法在推荐系统领域中的应用。

协同过滤算法较早被应用于推荐系统的研究，并且取得了一定的成功。文献[2]介绍了协同过滤算法的研究进展，并分类详细归纳了该算法所存在的问题及解决方案，最后提出了关于该算法在推荐系统领域中的研究热点。文献[3]以矩阵分解模型为基本模型，对 Adaboost 算法进行了改进并将其运用于推荐系统评分预测问题，通过引入阈值将评分预测问题转化为分类问题，实验结果表明该方法相比经典算法预测精度有所提高。文献[4]在贝叶斯矩阵分解（BMF）方法中引入辅助信息，其能够有效地提高推荐结果的准确性。

深度学习作为人工智能研究的一个热潮，许多学者已经尝试利用其强大的特征学习能力，将各种网络结构应用到推荐系统领域中。文献[5]对近几年基于深度学习的推荐系统的研究进展进行了综述，分析其与传统推荐系统的区别以及优势，并对其主要的研究方向、应用进展等进行了概括、比较和分析，最后对基于深度学习的推荐系统的未来发展趋势进行了分析和展望。文献[6]在 LSTM 的基础上增加了时间门，提出了一种新的 LSTM 模型（Time-LSTM）。与传统的 RNN 相比，Time-LSTM 能够捕获用户长短期的兴趣偏好，提高了推荐的性能。文献[7]利用深度学习模型从反馈文本中分别学习用户行为特征和项目特征。文献[8]为推荐系统提出了 word & wide 学习框架。文献[9]提出了基于产品的深度神经网络模型（PNN），以提高 DNN 在分类数据上的预测性能。

混合推荐算法，即融合了协同过滤算法与深度学习模型。文献[10]提出了一种通用的推荐框架 NCF，其利用网络结构代替 MF 的点积，取得了很好的推荐效果。文献[11]融合矩阵分解和 RNN 模型来探索用户和电影长短期特征，进一步利用海报信息来提高电影推荐系统的性能。文献[12]利用卷积神经网络捕获文本特征，并将其融入到 PMF 中，构成了 ConvMF 算法；同时还证明了该算法的有效性。文献[13]考虑到用户偏好和项目偏好的动态变化，将时间融入到 MF 的协同过滤算法中，构成了 TimeSVD＋＋算法；同时，利用 SADE 模型提取项目的内容特征来作为辅助信息，有效地解决了项目冷启动问题。文献[14]提出了一种协同深度学习（CDL）的分层贝叶斯模型，该模型将项目内容信息的深度表示特征与基于协同过滤算法的评分分解矩阵紧密结合。

3　融合动态协同过滤和深度学习的推荐算法

3.1　电影推荐中存在的问题

在基于 MF 的协同过滤算法的电影评分预测过程中，主要有以下因素对预测效果产生影响。

（1）电影评分矩阵的稀疏性。造成电影推荐系统中电影评分预测准确性不高的主要问题是数据的稀疏性问题。一般地，在电影推荐系统中有成千上万部电影，然而对于一个电影爱好者来说，他所评价过的电影不会超过所有电影的 2%，这样评分矩阵就会出现数据稀疏的问题，从而导致用户与用户之间、用户与项目之间存在的关系偏少，特征提取不精确，预测准确性较低。

（2）冷启动问题。冷启动问题主要分为两种：一种是新用户问题，另一种是新电影问题。在电影系统中，每天都会有成千上万的用户进行注册，同样，也有成千上万的新电影加入。对于新用户来说，如果他没有对系统中的任何电影做出评价，那么系统便无法通过分析得到用户的兴趣，这样就无法预测出该用户对电影的评分。同样，如果一部新的电影没有一个用户对它进行评价，则系统无法预测出某用户对该电影的评分。

（3）信息过期问题。用户兴趣偏好或者电影的受欢迎程度会随时间发生变化，而不是一成不变的。

本文针对上述问题，提出了融合动态协同过滤算法和深度学习模型的推荐算法。动态协同过滤算法是在 MF 的协同过滤算法中融入时间因素，以解决信息过期问题，并获取用户对电影评分行为在高维特征空间中的用户隐向量和电影隐向量。深度学习模型使用 CNN 捕获电影的文本特征，并使用

MLP 获取用户和电影特征在高维特征空间的隐向量。将深度学习模型学习到的隐向量融入到动态协同过滤算法中，以解决动态协同过滤算法中用户电影评分矩阵的稀疏性和冷启动问题。通过训练混合推荐算法，最小化评分预测值和真实值之间的误差，并在测试集上测试算法的预测效果。

3.2 动态协同过滤算法

动态协同过滤算法即 TimeSVD++算法[15]，是由最简单的 MF 逐步演变而成的，如图 1 所示。假设 N 个用户对 M 部电影的评分矩阵为 $\boldsymbol{r}\in\boldsymbol{R}^{N\times M}$，矩阵中的非空元素 r_{ui} 表示用户 u 对电影 i 的评分。

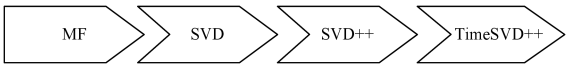

图 1 TimeSVD++的演变过程

Fig. 1 Evolution of TimeSVD++

在 MF 阶段，用户和电影的隐向量可分别表示为 $\boldsymbol{p}_u\in\boldsymbol{R}^{1\times F}$ 和 $\boldsymbol{q}_i\in\boldsymbol{R}^{F\times 1}$，其中 F 表示隐向量的维度。通过将 $\boldsymbol{p}_u$ 和 $\boldsymbol{q}_i$ 内积，得到评分矩阵 $\boldsymbol{r}$ 中元素 r_{ui} 的近似值 $\hat{r}_{ui}$，如式(1)所示；最小化 $\hat{r}_{ui}$ 与 r_{ui} 之间的误差，以得到最优的 $\boldsymbol{p}_u$ 和 $\boldsymbol{q}_i$，如式(2)所示。

$$\hat{r}_{ui}=\boldsymbol{p}_u\boldsymbol{q}_i \tag{1}$$

$$\min:Loss=\sum_{r_{ui}\neq 0}(r_{ui}-\hat{r}_{ui})^2 \tag{2}$$

在 SVD 阶段，只需在式(1)中加入偏置项：

$$\hat{r}_{ui}=\boldsymbol{p}_u\boldsymbol{q}_i+u+b_u+b_i \tag{3}$$

其中，u 表示训练集中所有评分的平均值；b_u 表示用户偏置，代表一个用户评分的平均值；b_i 表示电影偏置，代表一部电影被评分的平均值。

在 SVD++阶段，认为任何用户只要对电影 j 有过评分，无论评分多少，就已经在一定程度上反映了他对各个隐因子的喜好程度 $\boldsymbol{Y}_j=(\boldsymbol{Y}_{j1},\boldsymbol{Y}_{j2},\cdots,\boldsymbol{Y}_{jF})$，$\boldsymbol{Y}_j$ 是电影 j 所携带的隐特征。SVD++模型的评分预测函数如式(4)所示：

$$\hat{r}_{ui}=\left[\boldsymbol{p}_u+\frac{\sum_{j\in N(u)}\boldsymbol{Y}_j}{\sqrt{|N(u)|}}\right]\boldsymbol{q}_i+u+b_u+b_i \tag{4}$$

其中，$N(u)$ 表示用户 u 评价过的电影集合。

在 TimeSVD++阶段，认为电影的受欢迎程度是随时间变化的，用户的兴趣偏好也是随时间变化的，即 b_u、b_i 和 $\boldsymbol{p}_u$ 是随时间变化的，而电影特征的隐向量 $\boldsymbol{q}_i$ 是不随时间变化的。因此，TimeSVD++融入时间因素，将用户对电影的评分时间划分为若干个时间段，用 $Bin(t)$ 表示，处于同一时间段内的 b_u、b_i 和 $\boldsymbol{p}_u$ 是相同的，处于不同时间段的 b_u、b_i 和 $\boldsymbol{p}_u$ 是不同的。TimeSVD++模型的评分预测函数如式(5)所示。

$$\hat{r}_{ui}=\left[\boldsymbol{p}_u(t)+\frac{\sum_{j\in N(u)}\boldsymbol{Y}_j}{\sqrt{|N(u)|}}\right]\boldsymbol{q}_i+u+b_u(t)+b_i(t) \tag{5}$$

$$b_i(t)=b_i+b_{i,Bin(t)} \tag{6}$$

$$b_u(t)=b_u+\alpha_u\cdot dev_u(t)+b_{u,t} \tag{7}$$

$$dev_u(t)=\mathrm{sign}(t-t_u)\cdot|t-t_u|^{\beta} \tag{8}$$

$$\boldsymbol{p}_u(t)=(p_{u1}(t),p_{u2}(t),\cdots,p_{uF}(t)) \tag{9}$$

$$p_{uf}(t)=p_{uf}+\alpha_{uf}\cdot dev_u(t)+p_{uf,t},f=1,2,\cdots,F \tag{10}$$

式(6)中，$b_i(t)$ 表示电影 i 在 t 时间的偏置，由静态部分和动态部分组成；$b_{i,Bin(t)}$ 是电影 i 在 $Bin(t)$ 时间的偏置。式(7)中，$b_u(t)$ 表示用户 u 在 t 时间的偏置，由静态部分、用户兴趣偏好的逐渐转移和动态部分组成。式(8)中，t_u 表示所有评分时间的均值。式(9)中，$\boldsymbol{p}_u(t)$ 由用户 u 在 t 时间对各隐因子 f 的偏好组成。式(10)中，$p_{uf}(t)$ 表示用户 u 在 t 时间对各隐因子 f 的偏好，由静态部分、用户兴趣偏好的逐渐转移和动态部分组成。

TimeSVD++算法可通过式(11)最小化评分预测值与真实值之间的误差，在式(2)的基础上加入了参数的正则化，以防止模型过拟合。

$$\min:Loss=\sum_{r_{ui}\neq 0}(r_{ui}-\hat{r}_{ui})^2+\lambda[\|\boldsymbol{p}_u(t)\|^2+\|\boldsymbol{q}_i\|^2+\|b_u(t)\|^2+\|b_i(t)\|^2+\|\boldsymbol{Y}_j\|^2] \tag{11}$$

3.3 卷积神经网络和多层感知机

3.3.1 卷积神经网络

卷积神经网络已经被证实在文本建模方面具有强大的特征提取能力，能够获取句子的语义向量。卷积神经网络模型一般由输入层、卷积层、池化层和全连接层堆叠而成，如图 2 所示。第一层为输入层，输入预处理好的文本特征；第二层为卷积层，通过卷积核对输入层的词特征进行组合过滤，再使用激活函数进行计算，从而形成更抽象的特征模型；第三层为池化层，对上一层的词特征的相邻小区域进行聚类统计，从而得到新的特征；第四层为全连接层，输出最终的特征向量。

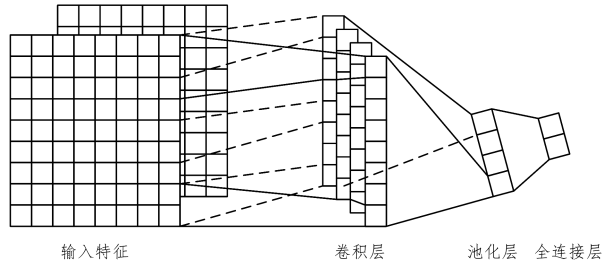

图 2 卷积神经网络模型

Fig. 2 Convolutional neural network model

3.3.2 多层感知机

多层感知机具有多个神经元层，第一层称为输入层，中间层称为隐层，最后一层称为输出层。MLP 根据需求定制隐层层数，并通过前向传播和反向传播来训练 MLP。最简单的 MLP 模型如图 3 所示，其只包含一层隐层。

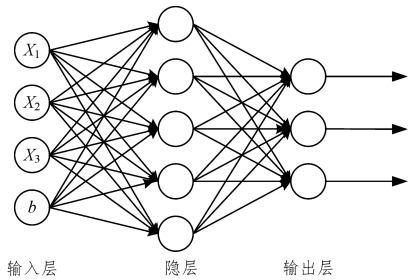

图 3 多层感知机模型

Fig. 3 Multi-layer perception model

3.4 混合推荐算法

本文提出的混合推荐算法简记为 CMLP-TimeSVD++，其整体框架如图 4 所示。

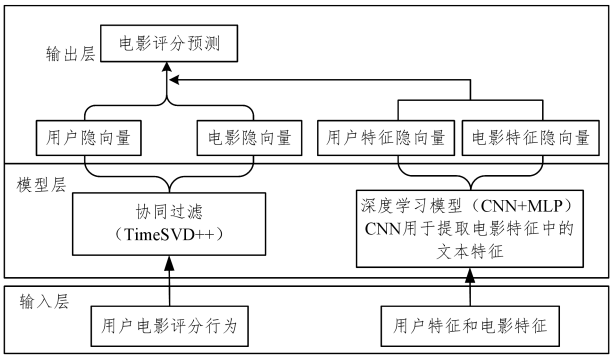

图 4　CMLP-TimeSVD++算法的整体框架

Fig. 4　Framework of CMLP-TimeSVD++ algorithm

本文以 MovieLens 为实验数据集，其涉及到的用户特征包含用户 ID、性别、年龄和职业，电影特征包含电影 ID、类别和标题。

3.4.1　用户特征隐向量

用户特征隐向量的学习包含以下 3 个步骤。

(1)将非数值特征转化为数值特征，即将性别中的男、女用 0 和 1 表示，职业中的 21 种职业分别用 0～20 表示。

(2)将用户特征输入到嵌入层，嵌入层的作用是将用户各特征转化为稠密向量。在嵌入层中，用户 ID、性别、年龄和职业的维度分别为 32，16，16 和 16。

(3)将嵌入层输出的用户的各特征向量输入到 MLP 中，得到最终的用户特征隐向量 $\boldsymbol{P}_u$。MLP 包含两层全连接层，第一层全接层有 128 个神经元，激活函数为修正线性单元(ReLU)，第二层全连接层有 200 个神经元，激活函数为 tanh。

3.4.2　电影特征隐向量

电影特征隐向量的学习包含以下 3 个步骤。

(1)1)将电影类别中 18 个类别映射成 0 到 17 个数字，由于每部电影可以对应至少 0 个类别，为了确保电影类别特征的维度(最多的类别个数加 1)一致性，增加一个特殊的填充符号(PAD)，其对应数字 18，用来补充不足 18 维度的电影类别。2)同样需要将电影标题的文本转化成数值，因此先收集所有标题的词语，共有 5 215 个，映射词语为 0 到 5 214 的数字；然后将标题表示成对应数字组成的向量，为了确保所有标题向量的长度(最长的标题词语个数加 1)一致，增加一个特殊的填充符号(PAD)，其对应数字 5 215，以补充不足 15 维度的电影标题。

(2)1)将电影 ID 和电影类别特征输入到嵌入层中，电影 ID 和电影类别的维度同为 32，并且将每部电影各类别对应的向量做加和操作，以代表这部电影的类别特征。2)将电影标题特征输入到 CNN 中，卷积层中使用的滑动窗口大小分别为 2，3，4 和 5，卷积核的大小为滑动窗口大小的 32 倍，共有 8 个卷积核，步长为 1；激活函数为 ReLU；池化层采用的是最大池化，池化大小为(15－滑动窗口大小＋1)＊1；全连接层采用了 Dropout 技术(在每一次训练学习网络时，该层随机屏蔽一些神经元，相当于把这些神经元从网络中“抹去”，以防止网络过拟合)，Dropout 为 0.5。

(3)1)将嵌入层输出的电影的各特征向量输入到 MLP 中，其包含 1 层全连接层，共有 64 个神经元，激活函数为修正线性单元(ReLU)。2)将第一个 MLP 中输出的特征向量和卷积神经网络输出的电影标题特征向量输入到第二个 MLP 中，得到最终的电影特征隐向量 $\boldsymbol{Q}_i$。该层包含 1 层全连接层，共有 200 个神经元，激活函数为 tanh。

3.4.3　CMLP-TimeSVD++算法

由图 4 可以看出，CMLP-TimeSVD++算法将用户特征隐向量 $\boldsymbol{P}_u$ 和电影特征隐向量 $\boldsymbol{Q}_i$ 融入到动态协同过滤的式(5)中：

$$\hat{r}_{ui}(t)=\left[\boldsymbol{p}_u(t)+\frac{\sum_{j\in N(u)}\boldsymbol{Y}_j}{\sqrt{|N(u)|}}\right]\boldsymbol{q}_i+u+b_u(t)+b_i(t)+\boldsymbol{P}_u\boldsymbol{Q}_i \tag{12}$$

其中，各参数的含义与 3.2 节中的相同。

该算法的损失函数在式(11)的基础上加以改进，如式(13)所示：

$$\begin{aligned}\min: Loss=&\sum_{r_{ui}\neq 0}(r_{ui}-\hat{r}_{ui})^2+\lambda[\|\boldsymbol{p}_u(t)\|^2+\|\boldsymbol{q}_i\|^2+\\&\|b_u(t)\|^2+\|b_i(t)\|^2+\|\boldsymbol{Y}_j\|^2+\|\boldsymbol{P}_u\|^2+\\&\|\boldsymbol{Q}_i\|^2]\end{aligned} \tag{13}$$

算法 1　CMLP-TimeSVD++

Input: user-to-movie rating matrix **r**, user features **P**, movie features **Q**, number of users N, number of movies M, latent factor number of user features F_1, latent factor number of movie features F_2, latent factor number of user-to-movie rating matrix F_3

Output: Mean_RMSE

Begin:

1. Dividing **r** into five training sets and test sets by five-fold cross validation.
2. hidden vector of user features **P**:
   P_D←Data preprocessing for **P**
   P_E←Entering **P_D** into the embedding layer MLP parameter initialization includes activation function, number of hidden layers and number of hidden layer neurons, etc.
   $\mathbf{P}_N$←Entering **P_E** into MLP
3. hidden vector of movie features **Q**:
   Q_D←Data preprocessing for **Q**
   Q_E←Entering the movie ID and movie categories in **Q_E** into the embedding layer CNN parameter initialization includes activation function, number of hidden layers, number of hidden layer neurons, convolution kernel and pooling layer, etc.
   Q_C←Entering the movie titles in **Q_E** into CNN MLP parameter initialization includes activation function, number of hidden layers and number of hidden layer neurons, etc.
   $\mathbf{Q}_M$←Entering **Q_C** and **Q_E** into MLP
4. CMLP-TimeSVD++ parameter initialization includes F_3, λ, β, epoch and activation function, etc.
5. RMSE_Sum←0
6. for test set in five test sets do
7.　$\hat{r}_{ui}$←Using formula (12) to calculate $\hat{r}_{ui}$
8.　loss←Using formula (13) to minimize loss
9.　RMSE←Using formula (14) to calculate RMSE
10.　RMSE_Sum←RMSE_Sum＋RMSE
11. end for
12. Mean_RMSE←RMSE_Sum/5

End

算法 1 给出了 CMLP-TimeSVD++算法的伪代码，其主要目标是通过迭代一定的次数 *epoch* 后，最小化预测值与真实值之间的误差，以 5 次实验的 *RMSE* 的平均值作为该算法的最终预测结果。算法的精度取决于算法中参数的设置，需要反复训练迭代及测试，从而选取最优的参数组合。在保证精度的同时，也要保证速度。算法的训练速度取决于每次训练迭代时更新权重使用的样本数 *Batch_Size*，当该数较大时，算法的精度相对较低，反之算法的精度较高。因此，需要在速度和精度之间进行折中，在保证精度的同时也能保证速度。

4 实验与分析结果

在基于 32GB 的 Ubuntu 14.04.5，Python3.6 及 Tensorflow1.3.0 环境下进行实验。实验主要涉及 3 个方面的内容：1)不同参数设置对 CMLP-TimeSVD++算法准确性的影响；2)基于最优参数的 CMLP-TimeSVD++算法，对比不同数据集的评分效果；3)基于最优参数的 CMLP-TimeSVD++算法，与其他学者的电影评分预测方法进行比较，从而证明本文方法的有效性。

4.1 数据集

实验所使用的数据集为美国明尼苏达大学 GroupLens 研究项目组所收集的 MovieLens 数据集中两个不同大小的数据集：MovieLens 100K 和 MovieLens 1M[16]。MovieLens 数据集包含用户数据、电影数据和评分数据，其中用户数据包含用户 ID、性别、年龄、职业和邮政编码，电影数据包含电影 ID、标题和类别，评分数据包含用户 ID、电影 ID、评分和时间戳，评分范围为 1～5。表 1 列出了这两个数据集的详细信息，提取了用户 ID、性别、年龄和职位作为用户特征，电影 ID、标题和类别作为电影特征。从表 1 可以看出，这两个数据集评分矩阵具有高稀疏性。

4.2 评测标准

本文以评分预测来度量推荐算法的准确度，以均方根误差(Root Mean Square Error，RMSE)为评估指标。RMSE 通过计算预测值与实际值之间的偏差来衡量推荐质量，其值越小，说明推荐算法的准确性越高。RMSE 的计算公式如下：

$$RMSE=\sqrt{\frac{\sum_{r_{ui}\in Test}(r_{ui}-\hat{r}_{ui})^2}{Num}} \tag{14}$$

其中，*Test* 表示测试集，r_{ui} 为测试集中用户 u 对电影 i 的实际评分，$\hat{r}_{ui}$ 为测试集中用户 u 对电影 i 的预测评分，*Num* 为测试集中用户评分的数量。

本文实验使用了五折交叉验证方法，选取其中 20%的评分数据作为测试集，剩下的作为训练集，一共进行 5 次实验，取这 5 次实验的 RMSE 的平均值作为最终结果的衡量指标。

4.3 比较算法

为了验证本文算法的有效性，将其与以下常见算法进行比较。

(1)ConvMF(Convolutional Matrix Factorization)[10]：融合卷积神经网络和 PMF。

(2)R-ConvMF(robust Convolutional Matrix Factorization)[15]：融合卷积神经网络、高斯噪声和 PMF。

(3)mSDA-CF(marginalized Stacked Denoising Auto-encoders based Collaborative Filtering)[14]：融合边缘降噪自编码器和 MF。

(4)Adaboost-PMF[3]：融合 Adaboost 算法和 PMF。

(5)CMLP-TimeSVD++：本文提出的方法，融合 CNN，MLP 和 TimeSVD++。

4.4 参数设置

为了得到最优参数组合下的 CMLP -TimeSVD+算法，需要对算法中的每一个参数选定候选范围，并对候选范围内的值进行训练及测试，从而确定最佳参数组合。但是，对如此多的参数进行调优，是一个极为耗时的工程，因此抓住主要的参数进行调优，而其他参数则借鉴其他学者的结果进行设置。

实验主要关注以下几个参数：1)隐因子数 F，候选范围为[5,10,15,20]；2)时间段的划分 $Bin(t)$，候选范围为[10,15,20,30]，其中的数值单位为天；3)学习率 lr，候选范围为[0.005,0.001,0.0005,0.0001]；4)每次训练迭代更新权重使用的样本数量 *Batch_Size*，对于 MovieLens 1M，候选范围为[128,256,512,1024]，对于 MovieLens 100K，候选范围为[32,64,128,256]。

表 1 实验数据集的详情信息

Table 1 Details of experimental datasets

Dataset	# Users	# Items	# Ratings	Sparsity/%	User Features	Item Features
MovieLens 100K	943	1682	100000	93.7	Age,Gender,Occupation	Title,Genres
MovieLens 1M	6040	3706	1000000	95.8	Age,Gender,Occupation	Title,Genres

深度学习模型中的参数设置可参见 3.4 节。参数的初始化操作使用的是[−0.1,0.1]之间的均匀分布函数。其余设置如下：$\lambda=0.02$，$\beta=0.4$，迭代次数 $epoch=10$，优化函数为 Adam。

4.5 实验结果与分析

4.5.1 超参数的实验结果与分析

根据 4.4 节所述的隐因子 F、时间段划分 $Bin(t)$、学习率 lr 和每次更新使用的样本数量 *Batch_Size*，对其候选范围分别进行实验，以分析不同参数取值对实验结果的影响。

(1)隐因子数 F

图 5 给出了表 1 中的两个数据集在不同隐因子数 F 下，CMLP -TimeSVD++算法的 RMSE 指标变化情况。

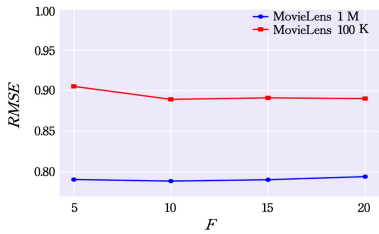

图 5 隐因子数 F 对 CMLP-TimeSVD++算法的影响

Fig. 5 Influence of F on CMLP-TimeSVD++ algorithm

由图 5 可以看出，当隐因子数 F 为 10 时，CMLP -Time

SVD++算法的 *RMSE* 达到最小，MovieLens 1 M 的 *RMSE* 为 0.788，MovieLens 100 K 的 *RMSE* 为 0.889。虽然隐因子数 F 为 5，15 和 20 时，其结果与隐因子数为 10 时的结果差别不大，但是综合考虑时间及效果后，取 CMLP -TimeSVD++算法的隐因子数 F 为 10。

(2)时间段划分 $Bin(t)$

图 6 给出了表 1 中的两个数据集在不同时间段划分 $Bin(t)$ 下，CMLP -TimeSVD++算法的 RMSE 指标变化情况。

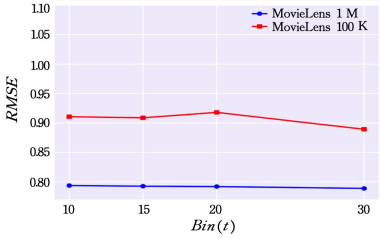

图 6　时间段划分 $Bin(t)$ 对 CMLP-TimeSVD++算法的影响

Fig. 6　Influence of $Bin(t)$ on CMLP-TimeSVD++ algorithm

由图 6 可以看出，两个数据集在 $Bin(t)$ 为 30 时，CMLP -TimeSVD++算法的 *RMSE* 达到最小，分别为 0.788 和 0.889。因此，CMLP -TimeSVD++算法的 $Bin(t)$ 为 30。

(3)学习率 lr

图 7 给出了表 1 中的两个数据集在不同学习率 lr 下，CMLP -TimeSVD++算法的 RMSE 指标变化情况。

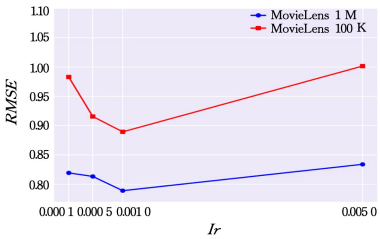

图 7　学习率 lr 对 CMLP-TimeSVD++算法的影响

Fig. 7　Influence of lr on CMLP-TimeSVD++ algorithm

由图 7 可以看出，两个数据集在学习率 lr 为 0.001 时，CMLP -TimeSVD++算法的 *RMSE* 达到最小，分别为 0.788 和 0.889。因此，CMLP -TimeSVD++算法的学习率 lr 为 0.001。

(4)每次训练迭代更新权重使用的样本数量 *Batch_Size*

图 8 给出了表 1 中的两个数据集在不同 *Batch_Size* 下，CMLP -TimeSVD++算法的 RMSE 指标变化情况。由于 MovieLens 1M 数据集的评分数据有 1000000 条，MovieLens-100 K 数据集的评分数据有 100 000 条，因此 MovieLens 1 M 的 *Batch_Size* 值比 MovieLens-100 K 大，其在加快迭代速度的同时，能够保证精度。

由图 8 可以看出，对于 MovieLens 1 M，当 *Batch_Size* 为 128 时，CMLP -TimeSVD++算法的 *RMSE* 达到最小，为 0.787，当 *Batch_Size* 为 256 时，*RMSE* 为 0.788，两者相差不大。但是，*Batch_Size* 为 128 时的训练时间是 *Batch_Size* 为 256 时的两倍，因此综合考虑时间及效果，CMLP-TimeSVD++算法的 *Batch_Size* 取 256。

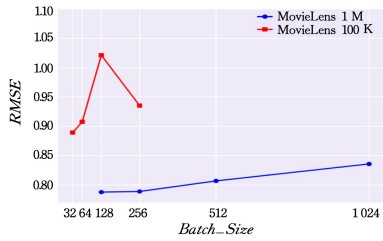

图 8　*Batch_Size* 对 CMLP-TimeSVD++算法的影响

Fig. 8　Influence of *Batch_Size* on CMLP-TimeSVD++ algorithm

对于 MovieLens 100 K，*Batch_Size* 为 32 时，CMLP-TimeSVD++算法的 *RMSE* 达到最小，为 0.889。由于MovieLens 100K 的评分数据相对较小，训练速度也比 MovieLens 1 M 快，因此 CMLP -TimeSVD++算法的 *Batch_Size* 为 32。

4.5.2　不同数据集的实验结果与分析

根据 4.5.1 节所述超参数的实验结果与分析，对于MovieLens 1 M 数据集，最优参数的组合如下：$F=10$，$Bin(t)=30$，$lr=0.001$，*Batch_Size*=256；对于 MovieLens 100 K 数据集，最优参数的组合如下：$F=10$，$Bin(t)=30$，$lr=0.001$，*Batch_Size* =32。对这两种数据集分别进行实验，以获取 CMLP -TimeSVD++算法在不同量级数据集上的实验结果，如表 2 所列。

表 2　不同数据集的的实验结果

Table 2　Experimental results for different datasets

Dataset	RMSE
MovieLens 1 M	0.788
MovieLens 100 K	0.889

4.5.3　不同算法的实验结果与分析

在表 1 中的数据集上，分别对 4.3 节所述的算法进行实验，以验证本文提出的 CMLP -TimeSVD++算法在最优参数下的电影评分预测准确率优于其他算法，结果如图 9 及表 3 所示。

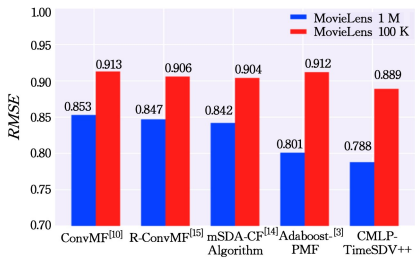

图 9　算法对比的实验结果

Fig. 9　Experimental results for algorithm comparison

表 3　算法对比的实验结果

Table 3　Experimental results for algorithm comparison

Algorithm	*RMSE*	
	MovieLens 1 M	MovieLens 100 K
ConvMF[12]	0.853	0.913
R-ConvMF[18]	0.847	0.906
mSDA-CF[17]	0.842	0.904
Adaboost-PMF[3]	0.801	0.912
CMLP-TimeSVD++	0.788	0.889

由图 9 及表 3 可以看出，以 RMSE 为指标，对于 MovieLens 1M 数据集，CMLP -TimeSVD++算法比最好的 Adaboost-PMF[3]算法提高了 1.623%；对于 MovieLens 100K 数据集，CMLP -TimeSVD++算法比最好的 mSDA-CF[14]算法提高了 1.659%。上述结果证明了本文提出的 CMLP-TimeSVD++算法在电影评分预测领域中具有更高的准确性。

CMLP-TimeSVD++算法融入了用户兴趣偏好特征、用户特征、电影特征以及时间因素，有效地解决了数据稀疏性和信息过期问题；同时在缺少用户兴趣偏好特征的情况下，可以借助用户特征和电影特征解决冷启动问题，进而提高了电影评分预测的准确性。

结束语 本文提出了 CMLP-TimeSVD++推荐算法，以电影评分预测为研究对象，其融合了动态协同过滤算法 TimeSVD++，CNN 和 MLP，以解决在电影评分推荐场景中的数据稀疏性、冷启动和信息过期问题。在 MovieLens 的两个数据集上分别进行了相关实验，结果表明，CMLP-TimeSVD++算法优于其他学者所提出的算法，提高了电影评分预测的准确性。但是，CMLP-TimeSVD++算法也存在缺点：1)算法复杂，可解释性差；2)算法中的参数较多，选取最优的参数组合需要花费较长的时间，而且参数的选取在一定程度上影响着算法的性能。

未来有以下问题值得进一步研究：1)CMLP-TimeSVD++算法的特征相对简单，可以被应用到其他场景中，以证明该算法的通用性；2)CMLP-TimeSVD++算法中文本特征的提取须进一步研究；3)针对时间特征，利用 LSTM 模型来获取时序关系。

参 考 文 献

[1] SUN H, HAN Z. An improved collaborative filtering algorithm for popular items of fusion items[J]. Miniature Microcomputer Systems, 2018, 39(4): 638-643. (in Chinese)
孙红，韩震. 融合物品热门因子的协同过滤改进算法[J]. 小型微型计算机系统，2018，39(4)：638-643.

[2] WENG X L, WANG Z J. Research progress of collaborative filtering recommendation algorithm[J]. Computer Engineering and Applications, 2018, 54(1): 25-31. (in Chinese)
翁小兰，王志坚. 协同过滤推荐算法研究进展[J]. 计算机工程与应用，2018，54(1)：25-31.

[3] XU R, ZHANG W. A recommendation system scoring prediction framework based on Adaboost algorithm[J]. Journal of Computer Systems, 2017, 26(8): 107-113. (in Chinese)
徐日，张谧. 基于 Adaboost 算法的推荐系统评分预测框架[J]. 计算机系统应用，2017，26(8)：107-113.

[4] PORTEOUS I, ASUNCION A, WELLING M. Bayesian matrix factorization with side information and dirichlet process mixtures [C]// Twenty-Fourth AAAI Conference on Artificial Intelligence. AAAI Press, 2010: 563-568.

[5] HUANG L W, JIANG B T, LU S Y, et al. A Survey of Recommendation Systems Based on Deep Learning [J]. Chinese Journal of Computers, 2018, 41(7): 191-219. (in Chinese)
黄立威，江碧涛，吕守业，等. 基于深度学习的推荐系统研究综述[J]. 计算机学报，2018，41(7)：191-219.

[6] ZHU Y, LI H, LIAO Y, et al. What to do next: modeling user behaviors by time-lstm [C]// Twenty-Sixth International Joint Conference on Artificial Intelligence. 2017: 3602-3608.

[7] ZHENG L, NOROOZI V, YU P S. Joint deep modeling of users and items using reviews for recommendation[C]// Proceedings of the Tenth ACM International Conference on Web Search and Data Mining. ACM, 2017: 425-434.

[8] CHENG H T, KOC L, HARMSEN J, et al. Wide & deep learning for recommender systems [C]// Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. ACM, 2016: 7-10.

[9] QU Y, CAI H, REN K, et al. Product-based neural networks for user response prediction [C]// 2016 IEEE 16th International Conference on Data Mining (ICDM). IEEE, 2016: 1149-1154.

[10] HE X, LIAO L, ZHANG H, et al. Neural collaborative filtering [C]// Proceedings of the 26th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2017: 173-182.

[11] ZHAO W, WANG W, YE J, et al. Leveraging long and short-term information in content-aware movie recommendation[J]. arXiv: 1712.09059, 2017.

[12] KIM D, PARK C, OH J, et al. Convolutional matrix factorization for document context-aware recommendation[C]// Proceedings of the 10th ACM Conference on Recommender Systems. ACM, 2016: 233-240.

[13] WEI J, HE J, CHEN K, et al. Collaborative filtering and deep learning based recommendation system for cold start items[J]. Expert Systems with Applications, 2017, 69: 29-39.

[14] WANG H, WANG N, YEUNG D Y. Collaborative deep learning for recommender systems[C]// Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2015: 1235-1244.

[15] KOREN Y. Collaborative filtering with temporal dynamics [C]// Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009: 447-456.

[16] HARPER F M, KONSTAN J A. The movieLens datasets[J]. Acm Transactions on Interactive Intelligent Systems, 2016, 5(4): 1-19.

[17] KAWALE J, KAWALE J, FU Y. Deep collaborative filtering via marginalized denoising auto-encoder [C]// ACM International on Conference on Information and Knowledge Management. ACM, 2015: 811-820.

[18] KIM D, PARK C, OH J, et al. Deep hybrid recommender systems via exploiting document context and statistics of items [J]. Information Sciences, 2017, 417(C): 72-87.