Recurrent Recommender Networks

accepted by WSDM'17，被引用次数：28，原文链接

摘要

推荐系统通常认为用户信息、商品的属性是静态的，而现实中存在时间动态性。当用户的喜好发生改变或者受到外部环境因素的影响，此时的特征都存在一定的偏差。本文提出的 **循环推荐网络** （Recurrent Recommender Networks，RRN）能够预测用户 **将来** 的行为轨迹，即通过 LSTM 刻画除了传统低阶矩阵分解的特征，更突出了动态的变化特征。这一模型是十分紧凑（参数规模更小）的，因为通过网络学习的目的不是隐藏状态，而只是状态转移函数。（注：目前还不能理解，请看后文。）

工作介绍

推荐系统理论、算法的研究也正在向实际应用靠拢，每当 Netflix 举办竞赛、测评，都会有许多新的方法出现，既有算法层面的，也有工程层面的。Netflix 提供的数据，是 tuple 的集合，一个 tuple 包括用户、电影、时间戳和打分，很自然地，任务定义为，给定用户、电影和时间，预测打分。性能评价则是打分的差距，通常用 RMSE、MSE 来衡量。常用的方法有：PMF，最近邻方法和一些聚类方法，从大数据中监督或非监督地学习一些用户特征，找到相似的用户或者物品进行推荐。

然而，这些方法都忽略了时间相关的信息和某些固有的随机情况，例如：

- 电影评价的改变：某一部电影刚上映，就被争议为史上最差，就会激发人们去看一看究竟有多差的好奇心，在这个时间后，会突然受到大众的狂热追捧。原本的烂片，就在大众的引导下成为受欢迎的大片，电影的特征必须做出调整来适应这种变化，**传统方法自始至终使用同样的特征，都忽略了这种反差改变。**
- 时间变化：圣诞节时期流行圣诞主题的影片，夏天时，有大海、冷饮的电影会比较受欢迎，而不是选择坐在围炉边讲故事的电影情节。虽然不是十分准确，但是这种时间规律上的变化也应该是要考虑的。
- 用户兴趣的变化：每个人在经历不同的事物后，心情会发生变化。例如突然沉迷某个导演，就会去看这个导演的所有片子；生活中遇到好事或是坏事，也会根据这种心情听对应主题的音乐等等。

需要指出的是，包括 Netflix 在内的测评和传统协同过滤方法，都违背了某种 **因果关系的先后要求**。过去，在训练过程中，通常都是打乱、随机选择一些交互记录，就会发生某些在未来的打分记录（在训练集中）会影响过去的打分预测（在预测集中），这种"后见之明"（hindsight）是不符合实际情况的。在这项工作中，遵守一种使用用户的当前状态和特征，预测将来喜好的前提。

本文提出的模型，能够对符合时间先后顺序的用户序列进行建模，对用户和物品通过隐变量描述特征，融合时间序列信息，具有动态的特性。

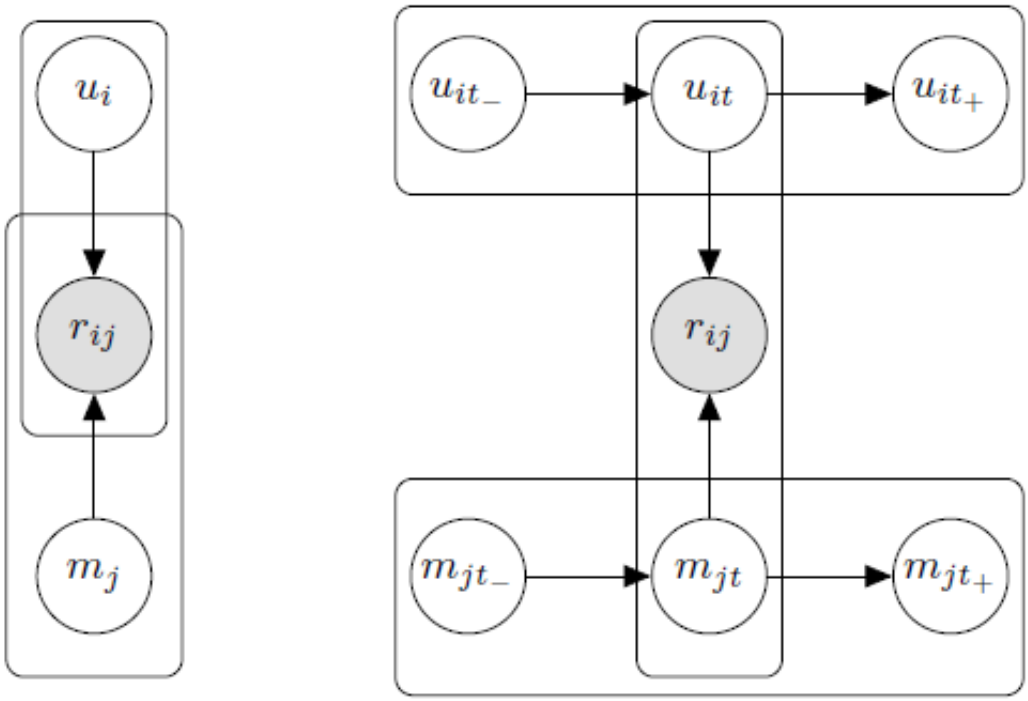


Figure 3: Left: time-independent recommendation model. User and movie parameters are stationary, ratings are drawn from $p(r_{ij}|u_i, m_j)$. Right: time-dependent recommendation model. Here both user and movie models follow a Markov chain and the ratings are drawn from $p(r_{ij}|u_{it}, m_{jt})$. Note that the plate notation is not entirely suitable for capturing the *additional* information of *which* (user,movie) pairs are rated and when.

从模型图中可见，对序列中的当前物品打分时，还与之前的物品和上一时刻用户的特征有关。这种做法和相关工作中的 TimeSVD++ 相比，不需要设计额外的时间相关参数，具备更好的扩展性。总结本文的主要贡献如下：

1. 非线性、非参数化的推荐系统模型：继 NCF、NNMF 之后，我们认识到用用户、商品特征向量的线性内积具有一定的局限性。（作者又说在这个模型下，非线性操作的实验效果并不十分明显，因此是不可捉摸的？？？）但本文是第一个动态刻画用户和商品特征的模型，并且是一个非参数化模型，不需要设计特定的用户状态空间；
2. 循环推荐网络：与典型的隐变量特征模型不同，这个网络中每次预测时状态都是动态计算得到的，不需要花费过多的计算能力去学习用户、商品的隐变量。
3. 实验设计：保证测试集中的用户记录都后于训练集中的记录出现，向实际情况靠拢，遵循时间上的先后因果关系。

相关工作

推荐系统和相关的算法

需要指出的是，基本的推荐系统算法几乎都不考虑时间先后。以打分预测这一任务为例，PMF（Probabilistic Matrix Factorization）是很通用的 benchmark，它的一些变体在特定的任务中也显示出不错的性能。考虑时间信息的，广受赞誉的 TimeSVD++ （2009 KDD） 通过一个独立的模型参数刻画时间信息，即增加一个与时间相关的偏置，可以说，这里的时间特征属于人工特征，借助了数据集相关的先验知识；此外，这一模型还违背了时间先后因果关系，将来的数据也被用来训练，去预测过去数据的打分。

和本文最接近的相关工作，是 AutoRec ，把矩阵分解视作一种编码操作。从用户的一系列打分信息中心，学习一个用户的低维、稠密向量表示，使得我们可以从最后的表示中恢复出用户的所有打分记录，是一个和 Encoder-Decoder 框架一致的过程，这也成为了神经网络模型在这项任务中的 state-of-the-art。

循环深度神经网络

RNN 和 LSTM，不再详细介绍。

模型

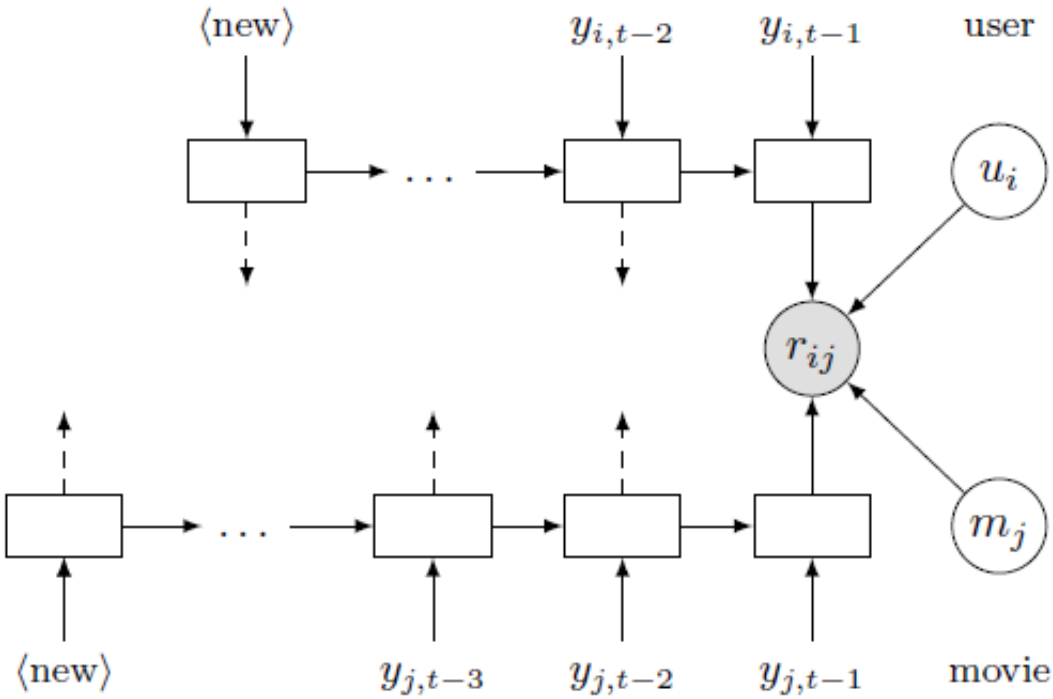


Figure 4: Recurrent Recommender Networks. We assume individual recurrent networks to address the temporal evolution of user and movie state respectively. The state evolution for a user depends on which movies (and how) a user rated previously. Likewise, a movie's parameters are dependent on the users that rated it in the previous time interval and its popularity among them. To capture stationary attributes we use an additional (conventional) set of auxiliary parameters u_i and m_j for users and movies respectively.

在 t 时刻预测用户 i 对商品 j 的打分，$\hat{r}_{ij|t} = f(u_{it}, m_{jt})$，这里的 $u_{i,t+1} = g(u_{it}, r_{ij|t})$，$m_{j,t+1} = h(m_{jt}, r_{ij|t})$。

用户、电影的状态

以用户为例，说明 RNN 的操作。令 $x_{tj} = k$ 表示用户在 t 时刻对电影 j 有打分 k，否则 $x_{tj} = 0$，从而 $x_t \in \mathbb{R}^M$ 表示用户在 t 时刻观测到的打分信息。此外，还定义了 τ_t 表示时间戳的数值（预处理成均值为0，单位方差），$1_{\text{newbie}} = 1$ 标记是否是新用户，上图中的 y_t 就定义为：

$$y_t := W_{\text{embed}}[x_t, 1_{\text{newbie}}, \tau_t, \tau_{t-1}]$$

y_t 是每一时刻在 LSTM 的输入层，通过 LSTM 操作，得到最终的用户特征 u_t：

$$u_t := \text{LSTM}(u_{t-1}, y_t)$$

值得一提的是，把时间戳的信息通过中心化的处理，直接“硬编码”在输入层中，能够表示用户在该时间内没有打分，并且隐含了这部电影的“年龄”，我认为还可以反映出不同用户对于每一个打分的时间间隔之间的差异性。

打分计算

尽管用户的喜好、电影的评价总是在变化中的，但还是存在一定的不变量，如用户的性别、电影的导演等信息。所以打分的计算还是包括了一些固定特征的考虑。

$$\hat{r}_{ij|t} = f(u_{it}, m_{jt}, u_i, m_j) := \langle \tilde{u}_{it}, \tilde{m}_{jt} \rangle + \langle u_i, m_j \rangle$$

这里的 $\tilde{u}_{it} = W_{\text{user}} u_{it} + b_{\text{user}}$ ， $\tilde{m}_{jt} = W_{\text{movie}} m_{jt} + b_{\text{movie}}$ 。（问题：为什么比直接用 $\langle u_{it}, m_{jt} \rangle$ 要好，是为了降维、特征压缩吗？）

模型优化与推理

与其他模型一样，最小化均方误差和正则惩罚项。由于有两个 LSTM，所以分先后对用户和电影的参数进行更新。

作者还特地解释了通过错误传播的更新能够影响之前的打分对当前打分的影响，在训练过程中其实很明朗。当用户看了一部满意的电影，那么对于他之后再看一部比较好的电影的打分可能会打折扣，要是遇到烂片，则烂片的打分会更低。

实验与分析

实验的数据集是 IMDb 和 Netflix 的 6 个月、1 年的子集和全集，未在训练集中出现的用户或者电影则从测试集中剔除。

实验设置（超参）

- 单层 LSTM，神经元个数设置为 40，输入层（y_t） embedding size = 40，20 维的动态状态特征（$\tilde{u}_{it}, \tilde{m}_{jt}$），静态特征 （$u_t, m_j$）维数 Netflix 设置为20，IMDb 为 160；
- 时间间隔的设置：Netflix 全集和 IMDb 中，x_t 与 x_{t-1} 用户和电影都间隔 2 个月，6个月的 Netflix 子集对用户间隔 1 天，对电影间隔 7 天。
- 基线对比实验：PMF， TimeSVD++，AutoRec（包括从用户和电影考虑的两个角度）

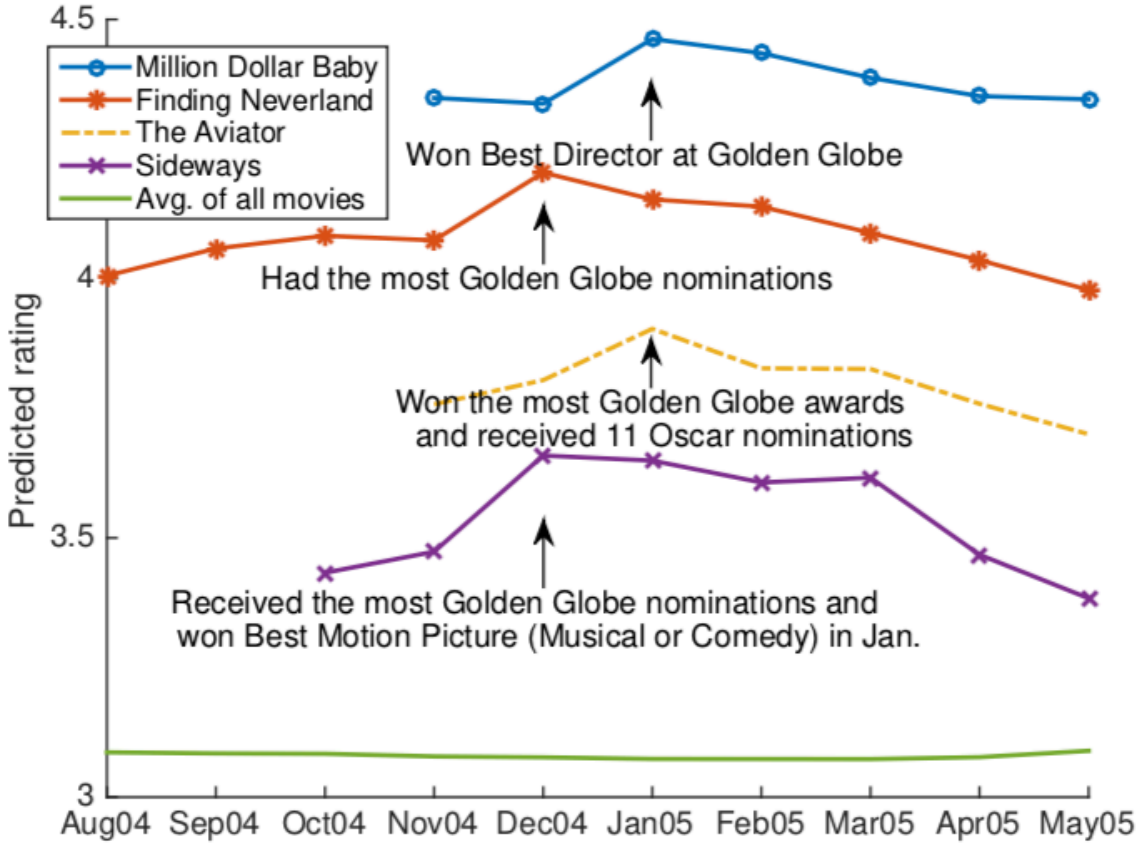
实验结果与分析

	PMF	I-AR	U-AR	T-SVD	RRN
IMDb	2.3913	2.0521	2.0290	2.0037	1.9703
Netflix 6 个月	0.9584	0.9778	0.9836	0.9589	0.9427
netflix 全集	0.9252	0.9364	0.9647	0.9275	0.9224

实验评估指标为 RMSE，可见 RRN 不仅在实验结果上比基线实验要好，而且需要的参数规模比其他的方法小 3-15倍。

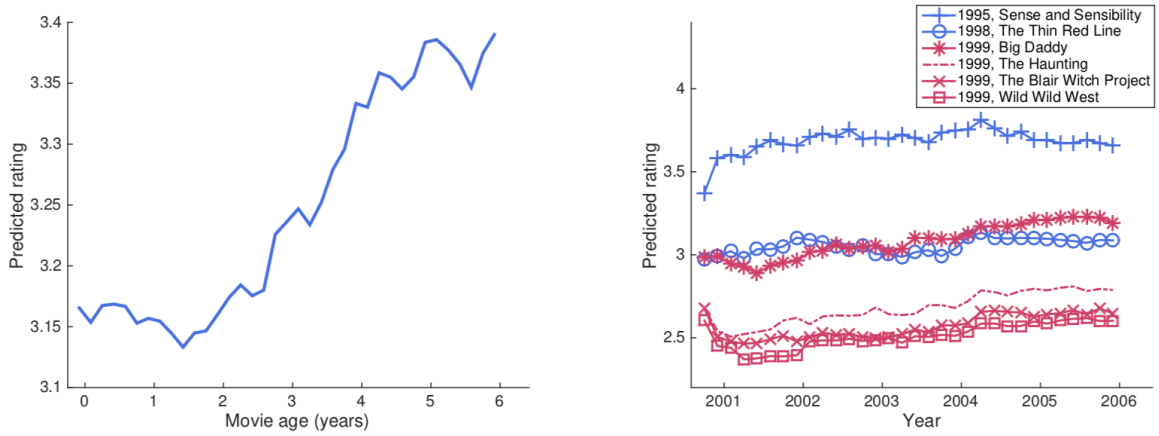
以下是一些 case study：

1. 电影的外在动态特征



在 Netflix 一年数据上的实验结果显示，一个月作为时间间隔，图中出现的几个预测打分突然升高，主要集中在电影节公布获奖之后。可见 RRN 模型能够反映出这一外部状况对用户打分的影响。

2. 电影内在动态特征



一方面，RRN 在预测电影打分的时候，显示出越古老的电影，预测值越高，是和整个数据集的走向一致的。另一方面，这种走向的变化不是通过类似 TimeSVD++ 或是什么时间衰减模型加上一个偏置项就可以做到的。

3. 用户打分区间的变化：数据集中，2003 年至2004 年，用户的平均打分模式发生了改变，作者说是评分网站提供的打分区间发生了变化，导致用户的打分尺度产生一个 spike。

关于冷启动

对于打分记录很少的用户和电影，RRN 比其他的方法分别提高了 2.5% 和 5.5%。这就体现出动态特征的优势了，因为静态特征学习的效果在这些冷启动用户或是电影上都很差。

增量训练

本文提出的 RRN 模型，按照一定的时间间隔更新用户、电影的动态特征和静态特征，当新的数据进来时只需要在之前的模型上继续训练即可，而不必从头开始训练，这也是模型的优势，可扩展，也提供了一种实时推荐系统的思想。（注：在新闻推荐系统中我就是这么设计的，对用户的特征向量用一个 RNN 不断更新）

时间间隔设计

越小的时间间隔，意味着模型的更新越频繁，且会导致更长的 RNN 输入序列，计算的代价就会越大，虽然性能可能会更好些。这里就存在一个 Trade-off 了。

结论与点评

本文的主要贡献，在【工作介绍】部分已经总结过了。本文第一次从用户和电影的角度同时利用 LSTM 进行动态的建模，这种动态的特征与数据集的统计特征是一致的，尤其是作者通过几个 case studies 的展示，增强了说服力。

回到我的近期工作上来，我对于用户的打分序列处理就太简单了，没有把时间戳这个数值融合进去，仅仅是根据时间戳去排序，而且由于 MovieLens 数据集中的用户打分时间戳在一段时间内都差别不大，所以切分是不太合理的？？？我现在觉得直接embedding 进去会更好一些。

其实我看到的 RNN 做序列化推荐的工作中，很少有增加 Attention 机制的，本文为了保留模型的动态、可增量化训练，可能舍弃了 Attention，我觉得在其他工作，例如预测购物篮中的下一个物品都是可以加的，因为可以体现某些商品对于特定的用户是十分重要的。

另一方面，我的师兄大佬曾经提出过十分类似的想法，当时不知道怎么处理时间间隔的问题，这篇文章的处理方式是一个很好的思路，因为我们当时的想法按照时间戳去划分一段时间的数据，直觉上认为产生的序列会很长，对于计算能力是一个很大的考验。虽然我认为本文的做法，用户在某个时刻 t 的打分信息 x_t 的维数是电影总数，感觉太大了...能不能想办法把这个维数降下来？有了这篇文章的工作，师兄提到的预测打分的同时，也预测一个时间戳，在实际生活中应该会更为有趣，不但能向你推荐物品，还会在特定的时刻推荐。