NAIS - Neural Attentive Item Similarity Model for Recommendation

Submitted to IEEE TKDE 2018，论文链接：https://www.comp.nus.edu.sg/~xiangnan/papers/neural-attentive-item-similarity-model.pdf

摘要

基于物品的协同过滤算法，在实际的推荐系统中更为常用。基于用户可能会喜欢相似、相关历史物品的假设，从而具有更好的解释性，在实时推荐的情境下，也可以进行高效地推荐。这种模型的关键在于如何衡量物品的相似度，较为传统的有余弦距离、皮尔逊相关系数，但是这种通用的方法并不一定适合推荐系统的场景。为此，一些工作会设计一个和推荐系统相关的损失函数，从用户、物品的角度来学习物品的隐藏表示。这些方法通常是线性的，所以本文提出一种基于注意力网络的非线性方法，能够获取到每个历史物品对用户当前喜好预测的重要性。NAIS 模型较 FISM（曾经的 state-of-the-art）有了性能上的提升，只是增加了一部分 attention 网络的参数规模。此外，NAIS 开创了神经网络对基于物品的协同过滤算法的时代，具有启发意义。

简介

协同过滤算法，以矩阵分解为代表，在推荐系统的打分预测任务上有十分良好的性能，但这种方法在实际推荐系统中很少被采用，用户-物品之间的协同过滤算法，在实时的推荐系统中，当用户有新的反馈时，很难对用户的隐藏特征进行实时学习或更新，每次都需要重新训练模型也是十分消耗资源与时间的。另一方面，通过物品-物品之间的系统过滤，通过向用户推荐历史记录中相似的物品，往往是一种保守、合理但是高效地方法，因此在业界更容易实现和接受。

简单的物品协同过滤方法，会使用一些简单的统计方法，如 Pearson 相关系数和余弦相似度来计算物品的相似性。这些方法比较死板，没有和推荐系统的特性和任务要求结合起来。相关工作中有通过物品-物品信息矩阵，设计了一个与推荐相关的损失函数来优化物品的特征学习，具有平方复杂度，当物品的数量庞大时，就不合适了。

为了解决这样的问题，Kabbur 等人提出了一种基于分解的学习模型，FISM，使用简单的向量内积来计算物品 embedding 之间的关系。FISM 的局限在于，认为用户的历史记录中的所有物品对于当前的预测都是等价的，显然近期的用户交互记录对于当前决策会有更大的影响。

本文提出了一种较 FISM 增强的物品相似性模型 NAIS，使得历史记录物品对用户的喜好有不同的重要性，主要是通过注意力机制结合神经表示学习来完成。但是标准的注意力机制很难奏效，因为用户的历史记录长短不一，NAIS 模型对此进行了平滑。最终 NAIS 相比 FISM 取得了 4.5% 的提高。

本文的代码开源在 https://github.com/AaronHeee/Neural-Attentive-Item-Similartiy-Model 。

背景与相关知识

标准 Item-based 协同过滤

标准做法中，认为用户 u 对物品 i 的预测得分由用户的历史记录中物品 $\mathcal{R}_u^+$得分的加权平均，权重即为待推荐物品与历史记录中的物品的相似度。这样的模型简单，优势在于物品之间的相似度提前计算好，那么在实时推荐时，只需要在 R_u^+ 中挑选出 top K 的物品，速度比较快。同时，即便有新的用户交互记录发生时，只需要更新那些和新添加的物品有较高相似度的那些，维护的代价也很小。

计算物品之间的相似度，一种方法是借助用户信息使用相似度函数来计算，较为新颖的方法有通过随机游走的方式来计算。

基于表示学习的模型

Ning 等人提出一种 SLIM 模型，使用了一个与推荐任务相关的损失函数来优化学习物品相似度矩阵 S ：

$$L = \frac{1}{2}\sum_{u=1}^{U}\sum_{i=1}^{I}(r_{ui} - \hat{y}_{ui})^2 + \beta\|S\|_2 + \gamma\|S\|_1$$

满足：$S \geq 0, diag(S) = 0$

这里对物品相似矩阵 S 进行了限制，L_1 正则项使得 S 比较稀疏，减少了物品与过多物品之间的相似度； L_2 正则项是为了防止过拟合；S 为非负矩阵，保证了相似性的要求，对角线为 0，避免了使用自身信息来预测自身的情况。

尽管 SLIM 模型能达到更高的准确率，但还存在以下两点缺陷：

- 离线的相似度矩阵计算 S 比较耗时
- 只能得到两个有共现过的物品（co-rated）之间的相似度，没有体现出物品相似的传递性

在此基础上提出的 FISM 模型，将物品表示为低维空间的向量，使用物品向量的内积作为相似度衡量，在预测时选用以下公式：

$$\hat{y}_{ui} = p_i^T \underbrace{(\frac{1}{|\mathcal{R}_u^+|^\alpha}\sum_{j\in\mathcal{R}_u^+\setminus\{i\}} q_j)}_{\text{user u's representation}}$$

在花括号中的可以视为用户 u 的一个表示，是一个对用户历史记录的平均，每个物品都是平等看待，显然是不合理的。FISM 模型没有使用打分信息，基于隐式反馈的方法。

NAIS 模型

模型设计

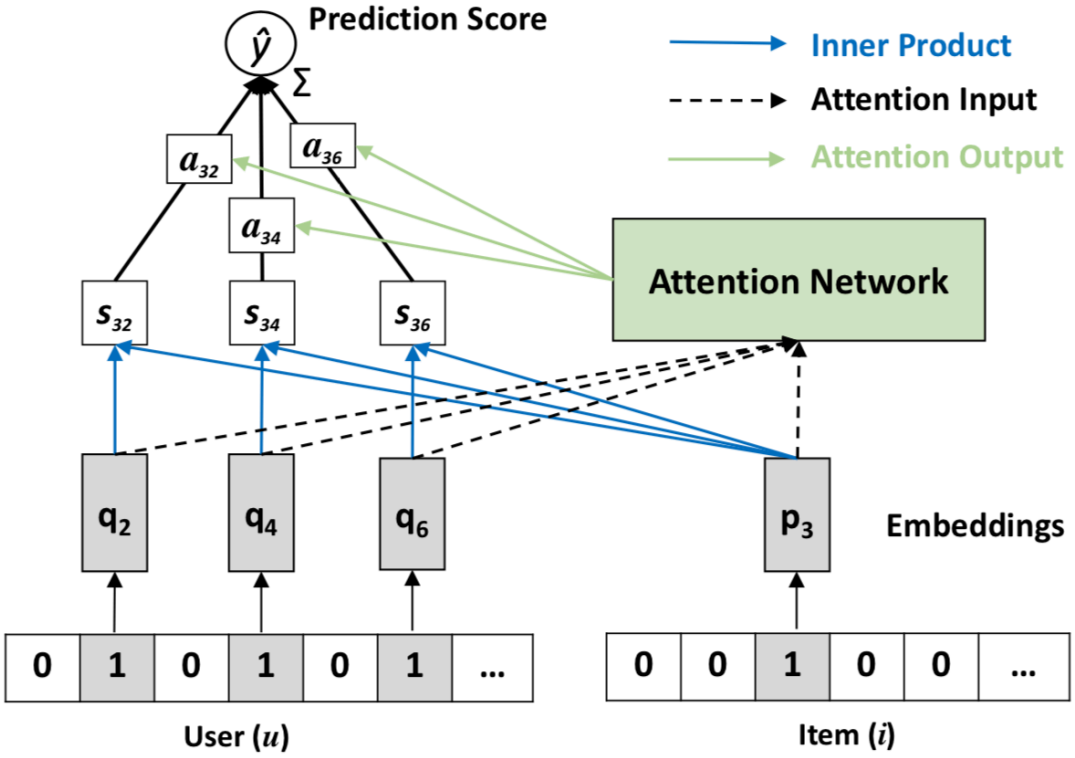

Fig. 1: The neural collaborative filtering framework of our Neural Attentive Item Similarity (NAIS) model.

核心思想是用户历史记录中的不同物品，应该对于候选物品的影响是不同的。

设计模式一

从 FISM 改变而来，设计打分为

$$\hat{y}_{ui} = p_i^T \underbrace{\left(\frac{1}{|\mathcal{R}_u^+|^\alpha} \sum_{j \in \mathcal{R}_u^+ \setminus \{i\}} a_j q_j\right)}_{\text{user u's representation}}$$

与 FISM 相比增加了一个 a_j 这一注意力模型可训练参数，可以控制物品 j 对当前物品 i 的贡献。但这样的设计太过于简单，因为每一个电影对于用户的权重影响，在每时每刻都是固定的。比如当前要预测一个用户对爱情电影的得分，那么相对于历史记录中的爱情电影 j，权重比较高。然而在预测恐怖电影的得分时，爱情电影的权重就不能还是那么高了。

设计模式二

改进就是，把 a_j 扩展到二维注意力矩阵 a_{ij} 表示物品 i 和 j 之间的注意力权重（贡献）。

$$\hat{y}_{ui} = p_i^T \underbrace{\left(\frac{1}{|\mathcal{R}_u^+|^\alpha} \sum_{j \in \mathcal{R}_u^+ \setminus \{i\}} a_{ij} q_j\right)}_{\text{user u's representation}}$$

这样还是存在一些隐患，当两个物品的配对没有在训练数据中出现，就无法学习到他们之间的注意力权重。

设计模式三

进一步的改进就是把 a_{ij} 抽象成一个函数计算得到，而不是简单的通过参数学习：$a_{ij} = f(p_i, q_j)$。

这样做的好处就是，即便物品 i 与 j 没有在训练数据中共现，只需要我们有可靠的学习到的物品隐藏表示 p_i、q_j 就能够计算出来。在神经网络和注意力机制中的通常做法，即通过 MLP 来建模，p_i 和 q_j 可以用拼接或者是点击的方式融合。

$$
\begin{cases}
1.\ f_{concat}(\mathbf{p}_i, \mathbf{q}_j) = \mathbf{h}^T ReLU(\mathbf{W} \begin{bmatrix} \mathbf{p}_i \\ \mathbf{q}_j \end{bmatrix} + \mathbf{b}) \\
2.\ f_{prod}(\mathbf{p}_i, \mathbf{q}_j) = \mathbf{h}^T ReLU(\mathbf{W}(\mathbf{p}_i \odot \mathbf{q}_j) + \mathbf{b})
\end{cases}
$$

$$
\hat{y}_{ui} = \mathbf{p}_i^T (\sum_{j \in \mathcal{R}_u^+ \setminus \{i\}} a_{ij} \mathbf{q}_j),
$$

$$
a_{ij} = \frac{\exp\left(f(\mathbf{p}_i, \mathbf{q}_j)\right)}{\sum_{j \in \mathcal{R}_u^+ \setminus \{i\}} \exp\left(f(\mathbf{p}_i, \mathbf{q}_j)\right)},
$$

至此，通过标准的 Attention 模型，就可以计算预测打分 $\hat{y}_{ui}$。

然而通过实验发现，这样的做法并没有取得预期的效果，低于 FISM，即使理论上是对 FISM 模型的泛化。作者借鉴了 CV、NLP 领域中的一些经验和技巧，认为原因主要在于：传统 Attention 层使用了 softmax 来归一化权重，比较接近 L_1 正则项那样进行了稀疏化，但这样的操作对于那些有很多历史记录的用户（活跃用户）来说，惩罚就太过于严重了。数据统计表明，在 MovieLens 和 Pinterest 数据集上的历史记录平均长度和方差分别是：(166, 37145) 和 (27, 57)，在 MovieLens 数据集上最长的历史记录为 2313。

NAIS 模型

考虑到上述情况，对基本的 a_{ij} 计算进行调整，对 softmax 操作进行平滑，减小对活跃用户的惩罚力度，并且降低了注意力权重的方差。

$$
\hat{y}_{ui} = \mathbf{p}_i^T (\sum_{j \in \mathcal{R}_u^+ \setminus \{i\}} a_{ij} \mathbf{q}_j),
$$

$$
a_{ij} = \frac{\exp\left(f(\mathbf{p}_i, \mathbf{q}_j)\right)}{\left[\sum_{j \in \mathcal{R}_u^+ \setminus \{i\}} \exp\left(f(\mathbf{p}_i, \mathbf{q}_j)\right)\right]^\beta},
$$

这里的 β 是一个 [0, 1] 的实数。尽管这样的操作会一定程度上使得注意力权重不是一个概率定义的形式，在结果上获得了大幅的提升。

此外，与 NCF 框架相比，NCF 的输入层用户的输入是根据 ID 进行 one-hot 编码，而 NAIS 模型的用户输入是 multi-hot 的输入。

模型优化

为了学习这些参数，需要设计一个与推荐系统比较相关的损失函数。与 NCF 类似，引入二分类的损失，分为正负两类 $\mathcal{R}^+$ 和 $\mathcal{R}^-$ 。

$$L = -\frac{1}{N}\left(\sum_{(u,i)\in\mathcal{R}^+}\log\sigma(\hat{y}_{ui}) + \sum_{(u,i)\in\mathcal{R}^-}\log(1-\sigma(\hat{y}_{ui}))\right) + \lambda\|\Theta\|^2$$

当然，有其他的损失函数，如 pair-wise，但是本文的重点在于注意力机制的应用，所以这部分会放在未来工作中。模型的实现中，借鉴了 BPR 的方式，对于每一个正样本，随机选择 4 个负样本，一起放入 batch 中训练。

预训练

从损失函数和参数空间上来看，很容易陷入局部最优，为了尽可能达到全局最优，需要一个较好的初始化。NAIS 模型首先通过 FISM 模型得到物品的 embedding。

模型分析

时间复杂性分析

相比于 FISM 模型，增加了 a_{ij} 的计算代价，但是也不需要在每一次预测时都重新计算，只需要计算一次保存起来，所以时间代价稍显增加。

个性化在线支持

在线场景下，用户的每一次交互都会作为流数据被采集，就需要实时更新用户的 top-K 推荐内容。立刻重新训练新模型显然不可能，只能采取增量更新的方式，可即便是参数的局部更新（只更新某个用户的 embedding），实际中也是很难做到的（可能还是需要重复学习的模型），同步问题也可能导致一些冲突。

NAIS 模型则不然，发生新的用户记录时，可直接动态更新用户的向量表示，因为基于物品的协同过滤方法使用的是用户的历史记录，而不是用户的 ID 来标记用户。在用户 u 对物品 t 有显式反馈后，预测用户对物品 i 的打分时，不需要从头计算 $\hat{y}_{ui}$，只需要计算 $a_{it}p_i^T q_t$，把这部分再加到原本的 $\hat{y}$ 中去。

注意力机制的选择

目前有拼接和内积两种方法，内积法的优势在于，attention 之后也是通过物品向量的内积 $p_i^T q_t$ 表示的，所以可能会有一定的同构促进作用；但是这样会丢失一些 p 和 q 本身的信息，这则是拼接方法的优势。

实验

实验设置

选用了和 NCF 一样的经过预处理的数据集 MovieLens 和 Pinterest（不活跃用户剔除、数据训练测试划分）。评价时选择了 leave-one-out 的做法——最后一个物品作为预测对象，同时随机采样 99 个负样本，最终在 100 个样本中排序，使用 Hit Ratio 和 NDCG 来评价。

基线实验有 Pop，itemKNN，FISM，MF-BPR，MF-eALS，MLP，没有选择 NeuMF，是因为它是一种把 NCF 和 MF 集成的方法。（难道是效果并不理想？）

实验的参数设置上，一开始并没有使用正则，直到发生过拟合再引入。先通过 FISM 模型得到物品表示的初始化，能略微提升模型的性能，加快模型的收敛速度。在实现细节上，由于 Tensorflow 中实现 Multi-hot 编码比较复杂，可以采用 masking trick，却很耗时。作者通过 mini-batch 的方法，随机只为一个用户进行采样和训练，不近速度更快，而且不需要为了 batch-size 调参。

注意力网络的有效性

NAIS 比 FISM 模型取得了更优异的性能。此外选择内积形式的 attention 函数更为适合，收敛更快。从量化指标上看，作者发现通过模型计算出的 a_{ij} 的方差在训练后期得到了扩大，符合每个物品对当前待推荐物品的贡献是不同的这一动机。原文中还给出了一个 case study，限于篇幅就不再叙述了。

预训练的作用在实验中体现出来的，可见深度学习、神经网络的学习，与模型的初始化还是有些关系的。

性能对比

TABLE 4: Recommendation accuracy scores (%) of compared methods at embedding size 16.

	MovieLens		Pinterest	
Methods	**HR**	**NDCG**	**HR**	**NDCG**
Pop	45.36	25.43	27.39	14.09
ItemKNN	62.27	35.87	78.57	48.32
MF-BPR	66.64	39.73	86.90	54.01
MF-eALS	67.88	39.83	87.13	52.55
MLP	68.41	41.03	86.48	53.85
FISM	66.47	39.49	87.40	55.22
NAIS-concat	**69.72**	**41.96**	**88.44**	**57.20**
NAIS-prod	**69.69**	**41.94**	**88.44**	**57.22**

- NAIS 模型比其他模型获得了较为显著的提高；
- 通过表示学习得到物品向量的方法（FISM）比一般简单的做法（itemKNN）有更大的性能提升，在推荐的任务上，还是要设计推荐相关的损失函数，体现了 task-based model；
- user-based CF 和 item-based CF 在两个数据集上各有千秋，可能 item-based 方法在稀疏数据集上性能要出色些。

超参设置

主要有三个超参：注意力隐藏层维数 a 和平滑参数 β 以及用于表示物品的 embedding size。

- 超参 a 在比较小时，NAIS 仍然优于 FISM，当 a 增大时，选择内积形式的优势比拼接的方式逐渐缩小；
- 实验证明，$\beta = 0.8$ 是一个较好的经验值，如果 $\beta = 1$，不使用平滑，性能剧降；
- embedding size 较小时，MLP 性能最好；增大后，NAIS 模型的优势渐渐明显，取 64 比较合理。

结论

本文提出了一种 item-based 的应用神经网络结构进行协同过滤的推荐模型，出于用户的历史消费物品对当前的候选物品的影响和相关性是不等同的。本文从 FISM 模型的表示学习开始，一步步加入、优化注意力机制模块，增加了一些 tricks，使之能够更好地从数据中建立一个合适的模型，通过实验不断验证猜想。

本文首次将神经网络融入 item-based CF，是未来相关工作的先驱。未来会考虑使用其他的深度模型来提升 NAIS，本文的设计比较简朴，也是出于模型的计算速度快，便于在工业界中应用。目前来看，NAIS 无法建模物品之间的二阶相似性——虽然两个物品没有直接成对出现在用户的历史记录中，但是如果它们都与某个物品十分相似，那么应该认为它们之间有相关性（体现传递的特性，Random Walk 来一发）。此外，在还可以引入卷积网络或者其他的网络结构来计算打分，因为许多相关的实验都体现出复杂网络优秀的学习能力。最后，这也是一次对推荐系统中基于深度学习算法的可解释性的一次探索。

个人见解：很多的 future work 都是空话，这里讲的还是不错的。推荐算法，尤其是深度学习引入后，可解释性比较欠缺，而 CF 方法有较好的可解释性和动机，简单套用 CV、NLP 相似模型的方法不一定行得通，作者在本文中也做了很多的改进和 trick 才最终全面打败了其他模型。

本文书写的逻辑是很好的，这也是何向南团队文章的优势，比较容易理解是一大特点。这点值得学习。此外，他在评价 hit ratio 的时候仅仅选择了其他 99 个随机的负样本来测试，似乎有点投机取巧，我决定在我的实验中也试一试这样的方式，与全局所有可能的候选物品推荐来计算 HR，真的是太难了……