论文｜被“玩烂”了的协同过滤加上神经网络怎么搞?

原创 Thinkgamer 搜索与推荐Wiki 2020-03-23

收录于话题
#论文笔记

19个

相信熟悉推荐系统的同学对于协同过滤（Collaborative Filtering）已经熟悉的不能再熟悉了，我也相信很多人心里在想“这么简单的协同，都2020年了，谁还用呀”。

俗话说得好，人不可貌相，海水不可斗量！CF作为最早的推荐算法，基于CF的改进在学术界和工业界应用的十分广泛，就在之前介绍的一篇论文里，介绍了腾讯实时ItemCF的实现和应用，所以说可千万别小瞧协同过滤了。

本篇论文主要介绍新加坡国立大学在17年发表的论文：Neural Collaborative Filtering（神经网络协同过滤），这篇论文主要介绍的点有：

- 协同过滤优化技巧中矩阵分解（MF）的限制
- 如何基于神经网络优化协同过滤

背景

在信息爆炸时代，推荐系统扮演着及其重要的角色，而且在很多在线服务上也应用的十分广泛，比如：电商网站、内容网站、社会媒体网站等。个性化推荐系统的关键是如何基于用户过去一段时间对物品的行为信息进行建模，最著名的属协同过滤（Collaborative Filtering，CF），在各式各样的CF技术中，矩阵分解（Matrix factorzation，MF）是应用的最广泛的。MF

将用户和物品映射到同一个潜在的空间中，用一个潜在的向量特征来表示一个用户或者物品，然后将用户与物品的向量内积来表示用户对物品的喜好。

尽管MF对于协同过滤非常有效，但是通过简单的内积计算会影响MF的效果。例如对于显式反馈的评分预测任务，可以通过合并用户和物品的评分偏差来改善MF的内积计算。对于内积计算操作而言，这可能只是一个很小的改动，但它正向的指出了设计一个更好的、专用的交互方式来对用户和物品的行为进行建模。仅仅使用线性相乘的内积计算不足以捕获复杂的用户交互行为。

尽管现在DNN在推荐中应用的比较广泛，但大多数是用户来进行预测建模，在协同过滤方面，仍基于MF来进行计算。本论文主要是利用神经网络来解决MF在协同过滤中的限制，而且论文基于的是用户的隐式反馈数据，与显式反馈相比，隐式反馈可以更加容易的收集用户数据，但也充满了挑战性，因为无法真实的反馈用户满意程度，而且缺乏一些自然的反馈数据。

本篇论文主要探索了如何基于对充满噪声的隐式反馈数据进行建模，主要贡献有以下三点：

- 提出了一种神经网络架构来对用户和物品的潜在特征进行建模，并为基于神经网络的协同过滤设计了通用的框架NCF
- 证明了MF可以被解释为NCF的一个特例，并利用多层感知器增加NCF模型的高度非线性。
- 基于两个真实的数据集证实了NCF的有效性

基础知识

1、了解隐式反馈数据

用M 、 N表示用户和物品的数目，用$Y \in R^{M*N}$表示用户对物品的隐式反馈，如果用户u对物品i有行为，则$y_{ui}=1$，否则为0。$y_{ui}=1$表示用户与物品之间存在交互，但这并不意味着用户u喜欢物品i，同样$y_{ui}=0$也不能表示用户u不喜欢物品i，可能是用户u并没有注意到物品i。

使用隐式反馈数据的推荐问题被形式化的用来估计用户u对物品i的得分，继而使用这些得分进行排序。基于模型的方法则认为数据可以通过一个底层模型来描述，通常这个底层模型被抽象的认为是$\hat{y}_{ui}=f(u,i|\Theta)$，$\hat{y}_{ui}$表示预测分，$\Theta$表示模型的参数，$f$则表示将模型参数转换为预测分的一个映射。

为了计算模型的参数Θ，现有的一些方法遵循机器学习的规则-优化目标函数。在研究中，主要有两种目标函数：PairWise Loss和PointWise Loss。在PointWise 方式中，通常遵循的是回归框架，即最小化$\hat{y}_{u,i}$ 、 $y_{u,i}$之间的误差。为了处理数据缺失的问题，则将未观察到的条目视为负反馈数据或者将观察到的反馈中负面样例作为负反馈数据。PiarWise 则认为观察到的样例（正样本）应该比未观察到的样例（负样本）得分高。因此代替了pointwise中的最小化方式，pairwise优化的目标是最大化观察到的样例（正样本）和未观察到的样例（负样本）。

而本文提出的NCF框架则利用神经网络求解参数Θ，继而来估计$\hat{y}_{ui}$，支持pointwise 和 pairwise。

> 关于 pointwise、pairwise、listwise可以参考之前发的一篇文章：怎么理解基于机器学习“四大支柱”划分的学习排序方法

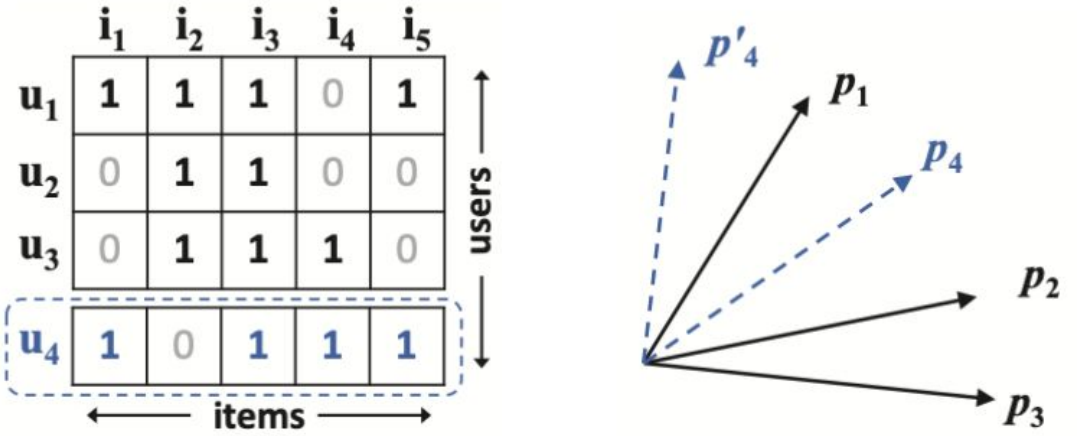
2、矩阵分解（MF）

MF将每个用户和物品的联系用潜在的向量特征来表示，用p_u和q_i分别表示用户和物品的隐含的向量，MF的计算方式是 p_u 、 q_i的内积，如下所示：

$$\hat{y}_{ui} = f(u, i|p_u, q_i) = p_u^T q_i = \sum_{k=1}^{K} p_{u,k} q_{i,k}$$

其中K表示隐含向量的长度。

如公式所示，假设用户和物品的每个维度的隐含特征都是彼此独立的，并以相同的权重进行线性组合，则MF是对用户和物品的隐含向量进行双向建模。从这个维度理解，MF可以看作是隐含向量的线性模型。



(a) user–item matrix (b) user latent space

Figure 1: An example illustrates MF's limitation. From data matrix (a), u_4 is most similar to u_1, followed by u_3, and lastly u_2. However in the latent space (b), placing p_4 closest to p_1 makes p_4 closer to p_2 than p_3, incurring a large ranking loss.

MF

上图展示了MF使用内积进行计算的限制，为了说明示例，有两个限制：

- 使用内积方式或者余弦相似度进行计算时，用户和物品的隐含向量特征维度必须一致
- 使用jaccard相似计算修正计算两个用户的相似度

从上图（a）可以看出，$s_{23}(0.66) > s_{12}(0.5) > s_{13}(0.4)$，即$u_2$ 、 u_3之间的相似度 > u_1 、 u_2之间的相似度 > u_1 、 u_3之间的相似度，可以将他们映射到图（b）所示的空间中，当一个新的用户u_4加入比较时，可以很容易的比较两两用户之间的相似度，$s_{41}(0.6) > s_{43}(0.4) > s_{41}(0.2)$，但是如果将$u_4$在图（b）中展示时，会发现和$p_4$最接近的顺序为：$p_1, p_2, p_3$，和我们计算的结果并不一致。

上图说明了，在低维场景下，使用内积方式的限制。一种解决办法是扩大隐含向量的维度，但是会影响模型的泛化能力，而且也会增大计算量。为了解决这个问题，论文中引入了DNN来解决内积方式的局限性。

神经网络协同过滤

1、NCF框架

a. 框架介绍

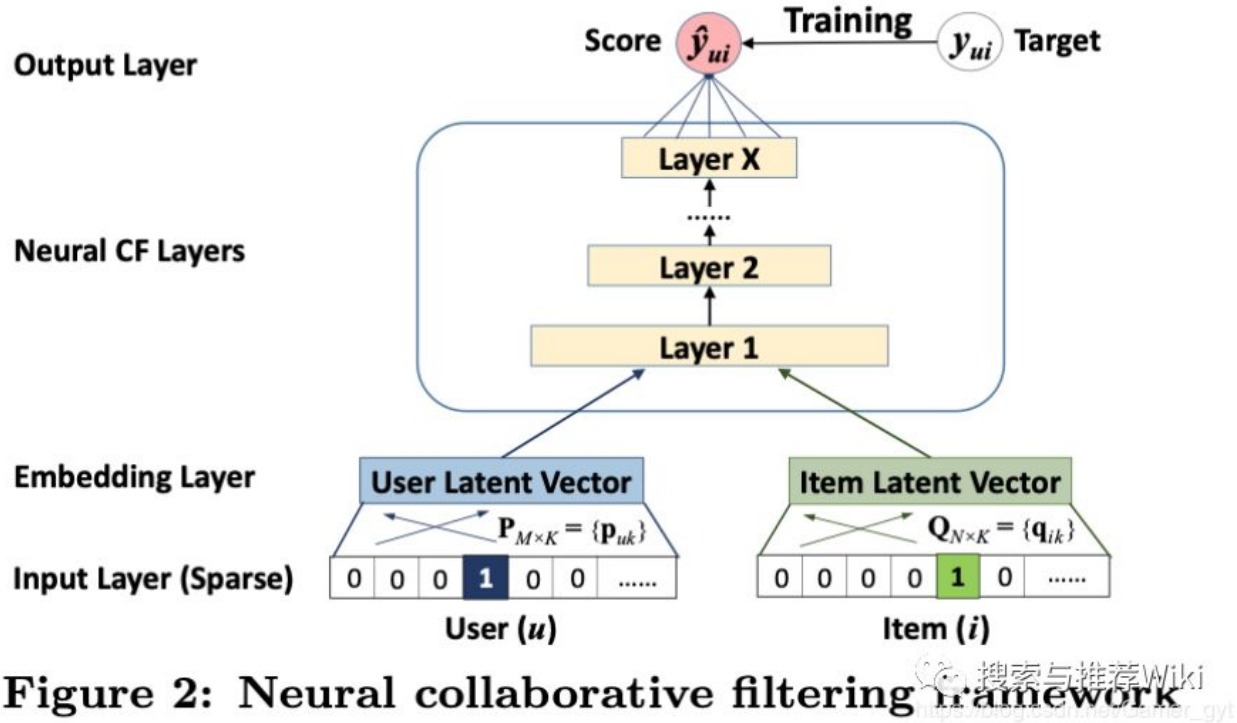


Figure 2: Neural collaborative filtering framework

NCF框架

一个通用的NCF框架如上图所示，最下面的input为v_u^U, v_i^I表示的是用户和物品的特征向量（这里是进行了ID的onehot编码），同时这里支持进行自定义继而对用户和物品进行建模，比如：上下文、基于内容、最近邻等。这种通用的内容输入，可以轻易的解决新用户的冷启动问题。

Input Layer输入的是稀疏向量，经过Embedding Layer转转化为稠密向量，之后向量被传入到多层的神经网络，这一整块被称为 Neural CF Layers，最后一层称为 Output Layer，输出层的维度决定了模型的功能，输出的$\hat{y}_{ui}$为预测分值，模型训练的目标是最小化$\hat{y}_{ui}$ 、 y_{ui}之间的误差，基于point wise方式的损失函数进行优化（关于point wise、pair wise、list wise的区别可以参考：怎么理解基于机器学习“四大支柱”划分的学习排序方法）。

定义NCF框架预测模型为：

$$\hat{y}_{u,i} = f(P^T v_u^U, Q^T v_i^I | P, Q, \Theta_f)$$

其中：

- P, Q表示用户和物品的隐向量
- Θ_f表示函数f的参数

函数 f被定义为多层神经网络，其表达式为：

$$f(P^T v_u^U, Q^T v_i^I) = \phi_{out}(\phi_X(\dots \phi_2(\phi_1(P^T v_u^U, Q^T v_i^I))\dots))$$

其中：

- ϕ_{out} 表示输出层
- ϕ_x 表示第x层，总共有X层

b. 参数学习

为了计算模型的参数，pointwise中通用的方法是平方损失（Squard loss），对应的表达式如下：

$$L_{sqr} = \sum_{(u,i) \in Y \cup Y^-} w_{u,i}(y_{ui} - \hat{y}_{ui})^2$$

其中：

- Y 表示正样本
- Y^- 表示负样本（可以是全部负样本，也可以是抽样后的）

虽然可以通过假设观测值是从高斯分布中生成的来解释平方损失函数，但对于隐式反馈数据可能无法很好的拟合，因此，论文中提出了一种基于概率的方法来学习 point wise方法的NCF，他能够很好的捕获隐式反馈数据。

对于输出的$\hat{y}_{ui}$需要将其限定在[0,1]，可以通过概率函数来实现（比如：Logistic 函数 或者 probit 函数）。

定义NCF的似然函数为：

$$p(Y,Y^-|P,Q,\Theta_f)=\prod_{(u,i)\in y}\hat{y}_{ui}\prod_{(u,i)\in y^-}(1-\hat{y}_{ui})$$

接着取负对数，可以得到最小化NCF的目标函数（参数求解可以通过SGD进行），如下所示：

$$L=-\sum_{(u,i)\in y}log\,\hat{y}_{ui}-\sum_{(u,i)\in y^-}log\,(1-\hat{y}_{ui})=-\sum_{(u,i)\in y\cup y^-}y_{ui}\,log\,\hat{y}_{ui}+(1-y_{ui})log\,(1-\hat{y}_{ui})$$

> 上述供水和交叉熵损失函数比较相似，也是log loss中的一种，这里通过对NCF进行概率函数映射，可以将隐式反馈转化为二进制分类解决推荐问题。

2、GMF（Generalized Matrix Factorization）

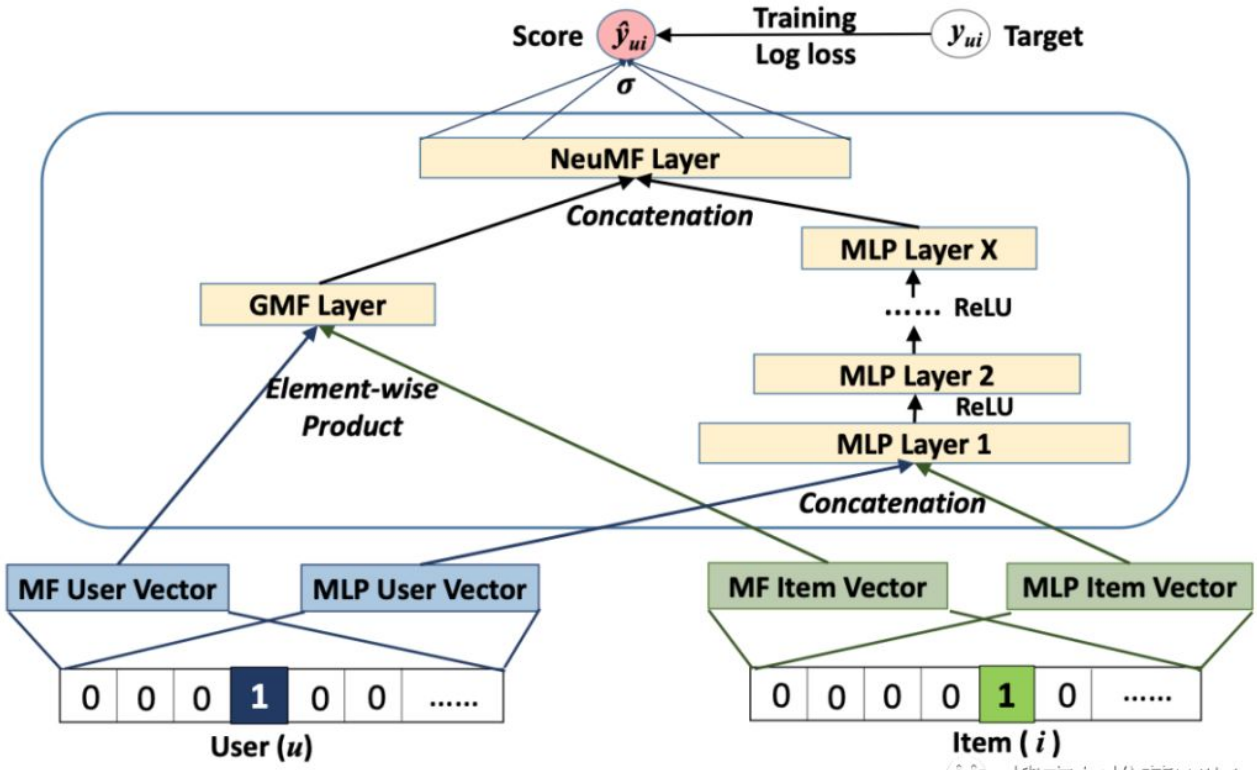


Figure 3: Neural matrix factorization model

NeuMF

GMF对应Figure 3 中的左半部分，用p_u表示用户的隐向量，用q_i表示物品的隐向量，可以定义GMF的函数为：

$$\phi_1(p_u,q_i)=p_u\odot q_i$$

⊙表示内积计算。将结果传到输出层，对应的计算公式为：

$$\hat{y}_{ui}=a_{out}(h^T(p_u\odot q_i))$$

其中：

- a_{out} 表示激活函数，本论文中用的是$\sigma(x)=1\,(1+e^{-x})$

- h 表示权重向量，使用 上文中的 log loss进行学习

3、MLP（Multi-Layer Perception）

简单的向量内积并不能捕获到用户和物品之间的交互信息，因此本论文通过使用MLP（Multi-Layer Perception）在连接的向量计算过程中增加隐藏层，以捕获用户和物品之间的交互，如Figure 3中的右半部分。

通过这种方式，可以赋予模型较大的灵活度和非线性，在NCF框架下MLP被定义为：

$$z_1 = \phi_1(p_u, q_i) = \begin{bmatrix} p_u \\ q_i \end{bmatrix}$$
$$\phi_2(z_1) = a_2(W_2^T z_1 + b_2)\dots$$
$$\phi_L(z_{L-1}) = a_L(W_L^T z_{L-1} + b_{L-1})$$
$$\hat{y}_{ui} = \sigma(h^T \phi_L(z_{L-1}))$$

MLP模型的激活函数选用的ReLU（还有sigmoid，tanh可以供选择），原因是：

- sigmoid 函数在横坐标接近正负无穷时，输出值接近 1或0，会导致神经元停止学习
- tanh 函数可以有sigmoid转化得到（$tanh(x/2) = 2\sigma(x) - 1$）,会造成同样的问题
- ReLU更符合生物学，且两端不饱和，同时ReLU非常适合稀疏函数，不会导致过拟合。

4、NeuMF（Neural Matrix Factorization）

a. NeuMF介绍

NeuMF则是GMF和MLP的结合，整体模型的思路如Figure 3所示，整体模型的表达式为：

$$\hat{y}_{ui} = \sigma(h^T a(p_u \odot q_i + W \begin{bmatrix} p_u \\ q_i \end{bmatrix} + b))$$

从图中可以看出，如果共享GMF和MLP的Embedding会限制模型融合的性能，比如必须保证相同长度的隐向量。

因此，GMF和MLP模型分开进行训练，在最后一层时进行连接，整体表达式可以表示为：

$$\phi^{GMF} = p_u^G \odot q_i^G$$
$$\phi^{MLP} = a_L(W_L^T a_{L-1}(\dots a_2(W_2^T \begin{bmatrix} p_u^M \\ q_i^M \end{bmatrix} + b_2)\dots)) + b_L$$
$$\hat{y}_{ui} = \sigma(h^T \begin{bmatrix} \phi^{GMF} & \phi^{MLP} \end{bmatrix})$$

MLP层采用的是ReLU激活函数。

b. 预训练

由于NeuMF目标函数是非凸的，所以基于梯度的优化方法只能找到局部最优解，而且初始化对于深度学习模型的收敛和性能有着重要的作用，因此这里采用的方法是先对GMF和MLP分开进行参数初始化，接着对两个模型进行训练，基于训练的结果初始化NeuMF的两部分参数。

$$h \leftarrow \begin{bmatrix} ah^{GMF} \\ (1-a)h^{MLP} \end{bmatrix}$$

优化方法采用的Adma。通过对频繁更新的参数进行较小的更新，对于不频繁更新的参数进行较大的更新，来调整参数的学习率。

实验

实验分为三部分进行，分别说明了以下三个问题：

- 基于NCF框架的方法是否优于最新的隐式协同过滤方法？
- 论文中提出的优化框架（带有负采样的对数损失）如何应用于推荐任务？
- MLP多层神经网络是否有利于用户和物品之间的行为交互？

答案肯定都是OK的，否则也不会发表成论文，哈哈 开玩笑！

实验所采用的数据集有两个：

- MovieLens
- Pinterest

数据集的描述如下：

Table 1: Statistics of the evaluation datasets.

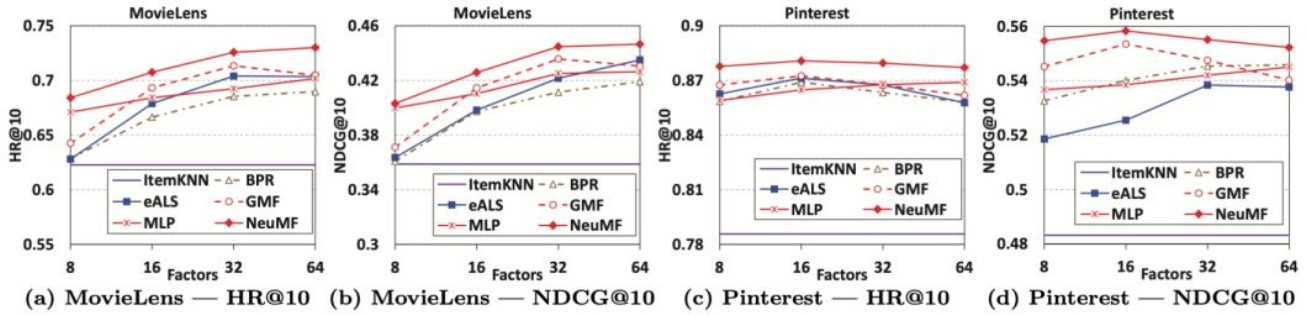
Dataset	Interaction#	Item#	User#	Sparsity
MovieLens	1,000,209	3,706	6,040	95.53%
Pinterest	1,500,809	9,916	55,187	99.73%

关于实验的技巧如下：

- 训练集和测试集的拆分，按照时间的顺序进行拆分，最接近当前时间的为测试集
- 随机选取训练集中用户没有行为的100个物品作为测试集中的物品，因为整体的物品太多，测试过程消耗资源和时间，不利于实验
- 每个评估指标取top 10物品进行评测
- 实验的指标：Hit Ratio、Nor- malized Discounted Cumulative Gain (NDCG）
- 实验采用的对比模型有：ItemPop、ItemKNN、BRP、eALS

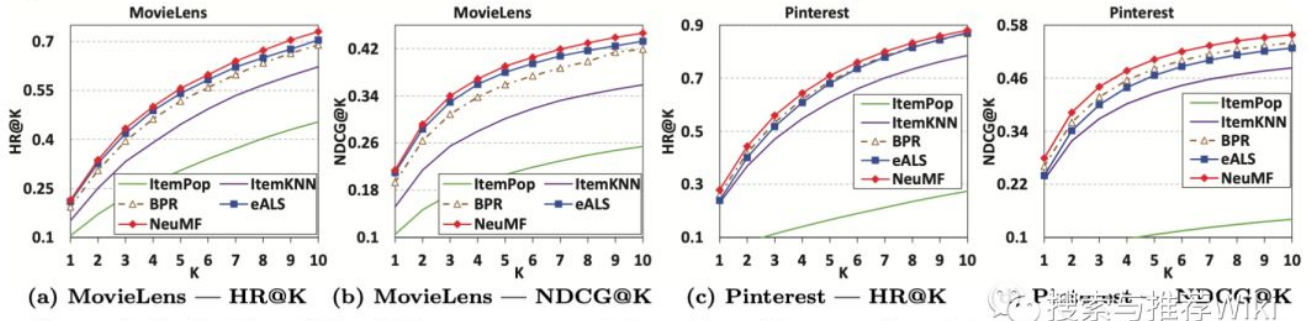
问题一的实验：

结果如下：



(a) MovieLens — HR@10 (b) MovieLens — NDCG@10 (c) Pinterest — HR@10 (d) Pinterest — NDCG@10

Figure 4: Performance of HR@10 and NDCG@10 *w.r.t.* the number of predictive factors on the two datasets.



(a) MovieLens — HR@K (b) MovieLens — NDCG@K (c) Pinterest — HR@K (d) Pinterest — NDCG@K

Figure 5: Evaluation of Top-K item recommendation where K ranges from 1 to 10 on the two datasets.

实验结果

下表则展示了有预训练GMF、MLP和没有预训练的结果对比：

Table 2: Performance of NeuMF with and without pre-training.

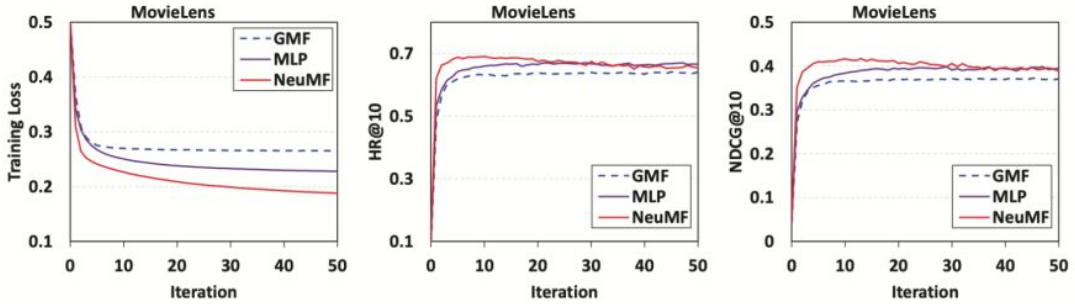
	With Pre-training		Without Pre-training	
Factors	**HR@10**	**NDCG@10**	**HR@10**	**NDCG@10**
MovieLens				
8	0.684	0.403	**0.688**	**0.410**
16	**0.707**	**0.426**	0.696	0.420
32	**0.726**	**0.445**	0.701	0.425
64	**0.730**	**0.447**	0.705	0.426
Pinterest				
8	**0.878**	**0.555**	0.869	0.546
16	**0.880**	**0.558**	0.871	0.547
32	**0.879**	**0.555**	0.870	0.549
64	**0.877**	**0.552**	0.872	0.548

是否有预训练对比

问题二的实验：

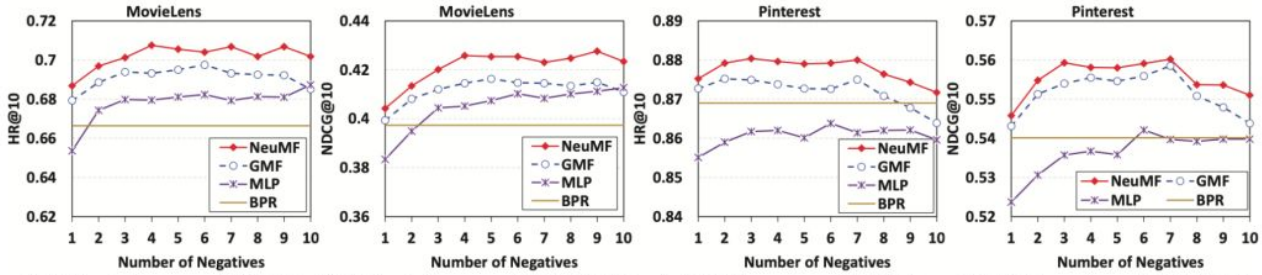
基于NCF框架进行的实验中主要是三种算法的对比：GMF 、MLP、NeuMF，在这三种算法中性能对比结果为：NeuMF > MLP > GMF，如Figure 6所示。

在Figure 7 中展示了不同的负采样率和评估指标之间的关系，每个正样本只有一个负样本不足以实现最佳性能，而对更多的负实例进行采样则是有益的。



(a) Training Loss (b) HR@10 (c) NDCG@10

Figure 6: Training loss and recommendation performance of NCF methods *w.r.t.* the number of iterations on MovieLens (factors=8).



(a) MovieLens — HR@10 (b) MovieLens — NDCG@10 (c) Pinterest — HR@10 (d) Pinterest — NDCG@10

Figure 7: Performance of NCF methods *w.r.t.* the number of negative samples p[illegible] (factors=16). The performance of BPR is also shown, which samples only one negative instance to pair with a positive instance for learning.

问题2实验

问题三的实验：

具有不同数量隐藏层的MLP对实验结果的影响如下：

Table 3: HR@10 of MLP with different layers.

Factors	MLP-0	MLP-1	MLP-2	MLP-3	MLP-4
MovieLens					
8	0.452	0.628	0.655	0.671	**0.678**
16	0.454	0.663	0.674	0.684	**0.690**
32	0.453	0.682	0.687	0.692	**0.699**
64	0.453	0.687	0.696	0.702	**0.707**
Pinterest					
8	0.275	0.848	0.855	0.859	**0.862**
16	0.274	0.855	0.861	0.865	**0.867**
32	0.273	0.861	0.863	**0.868**	0.867
64	0.274	0.864	0.867	0.869	**0.873**

Table 4: NDCG@10 of MLP with different layers.

Factors	MLP-0	MLP-1	MLP-2	MLP-3	MLP-4
MovieLens					
8	0.253	0.359	0.383	0.399	**0.406**
16	0.252	0.391	0.402	0.410	**0.415**
32	0.252	0.406	0.410	**0.425**	0.423
64	0.251	0.409	0.417	0.426	**0.432**
Pinterest					
8	0.141	0.526	0.534	0.536	**0.539**
16	0.141	0.532	0.536	0.538	**0.544**
32	0.142	0.537	0.538	0.542	**0.546**
64	0.141	0.538	0.542	0.54[illegible]	[illegible]

结果表明使用深度模型进行协作推荐的有效性。

至此，论文的内容已经阅读完成，关于文中算法的代码实现可以参考：《Neural Collaborative Filtering》NCF模型的理解以及python代码。

更多论文、报告、资源阅读请戳 “阅读原文”。