1 选题介绍

1.1 论文出处

题目：deep matrix factorization models for recommender systems

发表会议：2017 *International Joint Conferences on Artificial Intelligence* （IJCAI）

1.2 论文简介

作者将基于矩阵分解(Matrix Factorization)的推荐系统以输入为标准分为了两类，一种是基于隐式反馈(implicit feedback)为输入的推荐系统，另一种则是基于显式反馈(explicit feedback)为输入的推荐系统。隐式反馈输入往往是无法表达用户具体情感的数据，最常见的表达形式就是0-1矩阵，比如用户对物品的浏览情况，可以通过以用户为行，物品为列，构造一个0-1矩阵表达用户是否与物品存在交互。而显式反馈输入则往往比较明显了量化了用户的情感，最常见的就是评分矩阵，这种矩阵是通过以用户为行，物品为列，矩阵的元素为评分的形式构造。光从数字上看，一般隐式反馈比较适合与做提高召回率(recall)的模型输入。

由于推荐系统的数据集通常都是关于评分的，因此有一种比较特殊的隐式反馈输入，即将评分全都视为1，而这种变化会引起信息丢失的问题。 在这篇文章中，作者通过结合显式反馈和隐式反馈这两种输入，有效的解决了信息丢失的问题。不仅如此，作者还通过结合神经网络，提出了一种新型的矩阵分解模型，称为Deep Matrix Factorization (DMF)

本次论文中，我们把DMF的这种思想应用到传统的隐因子模型（latent factor model, LFM)中，先利用多层感知机来学习用户对物品的交互信息更深层次的非线性特征因子，提出了多层隐语义模型（multi latent factor model，MLFM）。接下来将特征因子的线性信息与非线性信息进行融合，提出了混合隐语义模型（fusion latent factor model，FLMF）。在两个真实数据集上的实验结果表明，本文提出的方法在Top-N推荐的性能方面要好于传统的隐语义模型，并且能够较好地处理稀疏数据集。最后我们还扩展测试了多组不同的实验参数对推荐效果的影响。

2 论文模型架构及算法

2.1 多层隐语义模型（MLFM）

LFM虽然可以有效利用隐式反馈来学习到用户和物品的隐语义表示；然而LFM学习到的往往是一种线性的浅层的特征。基于此，我们利用多层感知机来获得用户和物品更深层次的特征，提出了多层隐语义模型（multi latent factor model，MLFM），其网络结构如图2.1所示：

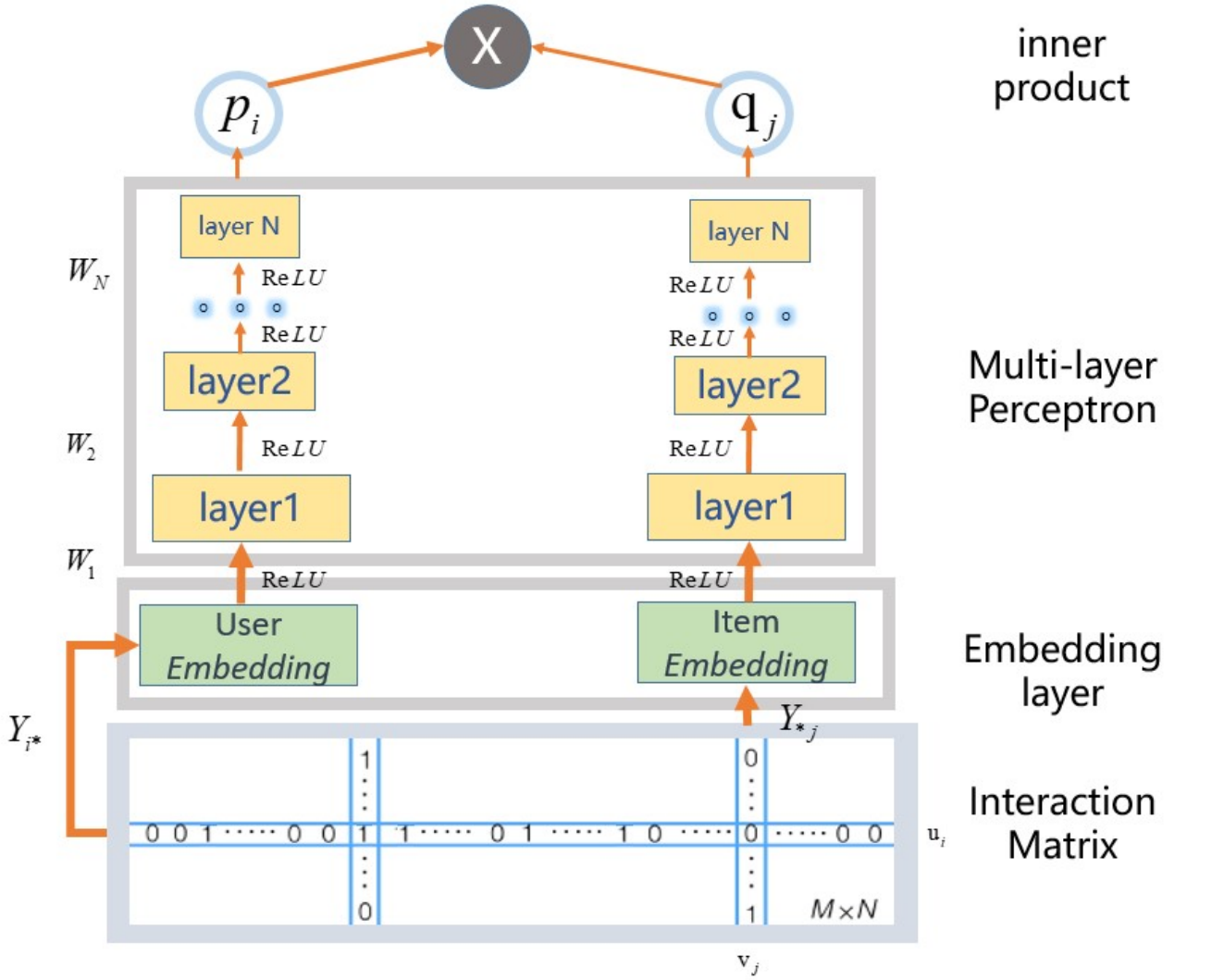


图2.1 MLFM模型架构

这里，我们使用之前得到的用户-物品矩阵Y作为输入，Y_{i*}是一个高维向量，表示的是用户u_i对所有物品的交互信息；而Y_{*j}则代表物品i_j与所有用户之间的交互信息，同样也是一个高维向量。MLFM首先利用和Y_{i*} 、Y_{*j} 得到用户嵌入层和物品嵌入层。之后利用多层感知机来得到用户和物品的隐语义表示。这里我们具体介绍一下多层感知机是如何应用到MLFM之中的。

我们用x来表示输入向量，用y来表示输出向量，用l_i来表示中间的隐藏层。并且用W_i来表i示第i个权重矩阵，用b_i来表示第i个偏置项，用h来表示最终的隐藏层表示。于是可以得到如下公式：

$$
\begin{aligned}
l_1 &= W_1 x \\
li &= f\left(W_{i-1} l_{i-1} + b_i\right), i = 2, \cdots, N-1 \\
h &= f\left(W_N I_{N-1} + b_N\right)
\end{aligned}
$$

这里我们使用ReLU来作为输出层和隐藏层l_i的激励函数,其中ReLU是一个线性分段函数，可用如下公式表示：

$$
\mathrm{ReL}\,U(x) = \begin{cases} x, & \text{if} \quad x > 0 \\ 0, & \text{if} \quad x \leq 0 \end{cases}
$$

因此由Y_i*和Y_{*j} 分别映射得到的低维隐语义表示p_i和q_j可以用如下公式表示：

$$p_i = f_N\left(\cdots f_3\left(W_{U2} f_2\left(Y_P W_{U1}\right)\right)\cdots\right)$$
$$q_j = f_N\left(\cdots f_3\left(W_{V2} f_2\left(Y_{*j}^T W_{V1}\right)\right)\cdots\right)$$

这里的$W_{Ul}(i = 1\cdots N-1)$和$W_{Ii}(i = 1\cdots N-1)$分别是用户和物品矩阵的权重。

而与LFM相似，最后的预测分数$\hat{Y}_{ij}$用公式表示为：

$$\hat{Y}_{ij} = F_{MFM}\left(u_i, v_j|\theta\right) = p_i^T q_j$$

与之前的LFM相同的是，MLFM也使用二值交叉熵来作为损失函数进行参数的优化。只不过与LFM得到的线性浅层隐语义不同，MLFM得到的p_i 和q_j则是用户和物品的非线性深层特征表示。

最后，将MLFM应用到本次试验的伪代码可以表示为：

Algorithm 3 MLFM

输入:

$Iter$: 训练的迭代次数,

neg_num: 每一个正样本所对应的负样本数量

R: 初始的打分矩阵

输出:

$W_{Ui}(i = 1\cdots N-1)$: 用户矩阵的权重

$W_{Vi}(i = 1\cdots N-1)$: 物品矩阵的权重

Initialisation:

set $Y \leftarrow$ use Equation 3.1 with R

set $Y^+ \leftarrow$ all none zero interactions in Y

set $Y^- \leftarrow$ all zero interaction in Y

set $Y^-_{sampled} \leftarrow$ sample neg_num interactions from Y^-

set $T \leftarrow Y^+ \bigcup Y^-_{sampled}$

for i from 1 to $Iter$ **do**

  for each iteraction of u_i and i_j in T **do**

    set $p_i, q_j \leftarrow$ with input of Y_{i*}, Y_{*j}

    set $\hat{Y}_{ij} \leftarrow$ use Equation 3.9 with input p_i, q_j

    set $L \leftarrow$ use Equation 3.5 with input $\hat{Y}_{ij}, Y_{ij}$

    use back propagation to optimize model paramaters

  end for

end for

2.2 混合隐语义模型（FLFM）

在前面的内容中，本文介绍了可以得到用户与物品线性浅层隐语义表示的传统LFM模型，以及加入了多层感知机，可以学习到用户与物品更深层非线性隐语义表示的MLFM模型。而接下来，我们将LFM算法以及MLFM算法进行融合，得到一种结合了用户与物品线性以及非线性特征的混合隐语义模型（fusion latent factor model，FLFM）。其网络架构图如图2.2所示：

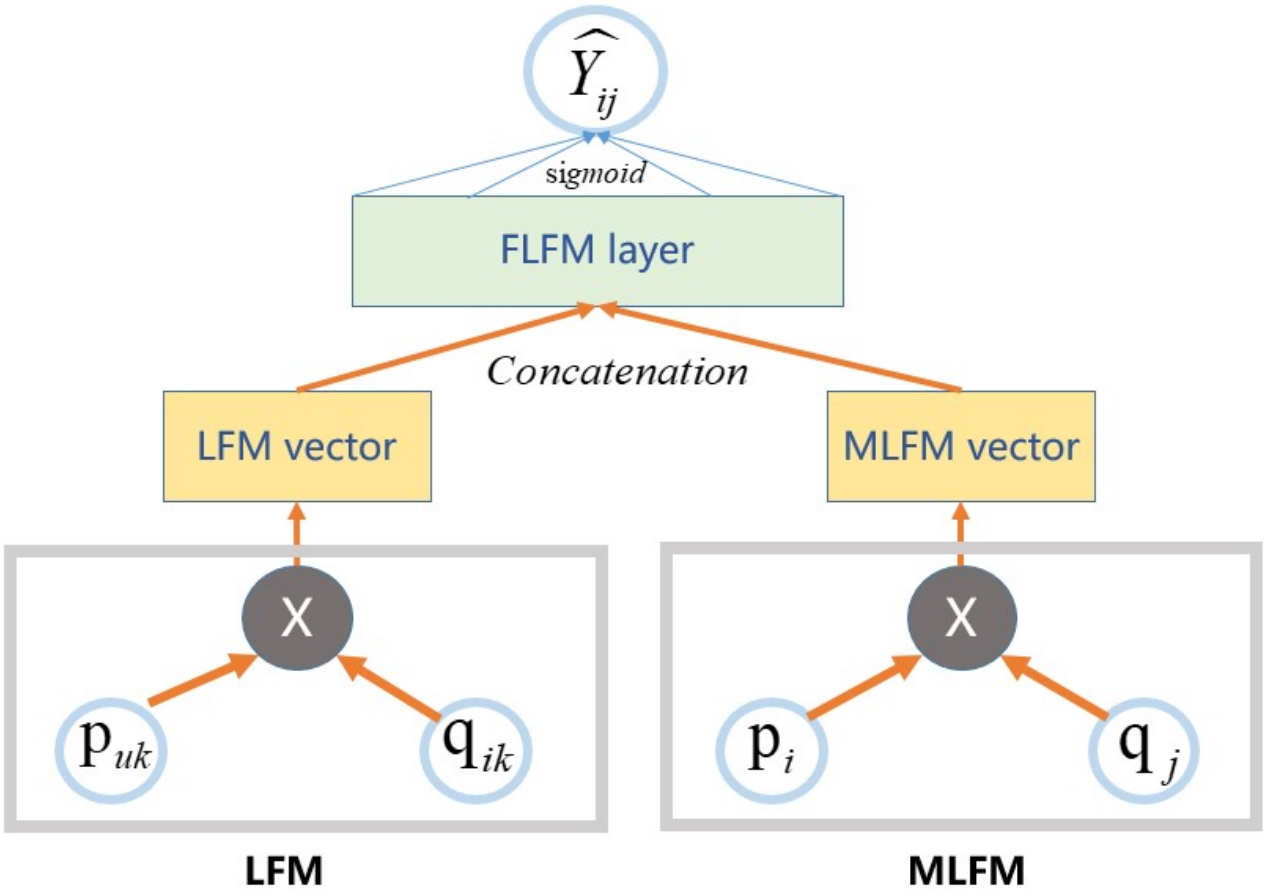


图2.2 FLFM模型架构

这里的$p_u k$和$q_i k$是利用LFM分别学习到的用户和物品线性隐语义，而p_i和则q_j 是利用MLFM分别学习到的用户和物品非线性隐语义。在FLFM模型的实现中，我们将乘积之后得到的线性特征向量与非线性特征向量进行连接操作，并使用sigmoid作为激励函数，来得到最终的预测分数$\hat{Y}_{ij}$。其实现原理用公式表示如下：

$$\hat{Y}_{ij} = F_{FLFM}\left(u_i, v_j|\theta\right) = \sigma\left(f\left(p_{uk}^T q_{ik} \oplus p_i^T q_j\right)\right)$$

其中的$\oplus$是连接操作，而σ则代表sigmoid激励函数，其常应用于二分类问题中，并将结果映射到[0,1]范围内，其数学表达式如下：

$$f(x) = \frac{1}{1+e^{-x}}$$

而与前面的模型实现一样，FLFM也使用二值交叉熵来作为损失函数进行参数的优化这样，将FLFM应用到本次试验中，就可以用如下的伪代码来表示：

Algorithm 4 FLFM

输入:

$Iter$: 训练的迭代次数,

neg_num: 每一个正样本所对应的负样本数量

R: 初始的打分矩阵

输出:

$W_{Ui}(i = 1 \cdots N-1)$: 用户矩阵的权重

$W_{Vi}(i = 1 \cdots N-1)$: 物品矩阵的权重

Initialisation:

set $Y \leftarrow$ use Equation 3.1 with R

set $Y^+ \leftarrow$ all none zero interactions in Y

set $Y^- \leftarrow$ all zero interaction in Y

set $Y^-_{sampled} \leftarrow$ sample neg_num interactions from Y^-

set $T \leftarrow Y^+ \bigcup Y^-_{sampled}$

Initialize p_{uk}, q_{jk} with Gaussian distribution

for i from 1 to $Iter$ **do**

  for each iteraction of u_i and i_j in T **do**

    set p_i, $q_j \leftarrow$ with input of Y_{i*}, Y_{*j}

    set $vector_{LFM} \leftarrow$ use Equation 3.3 with input of p_{uk} q_{jk}

    set $vector_{MLFM} \leftarrow$ use Equation 3.9 with input p_i, q_j

    set $vector_{FLFM} \leftarrow vector_{LFM} \bowtie vector_{FLFM}$

    set $\hat{Y}_{ij} \leftarrow$ use Equation 3.10 with input $vector_{FLFM}$

    set $L \leftarrow$ use Equation 3.5 with input $\hat{Y}_{ij}$, Y_{ij}

    use back propagation to optimize model paramaters

  end for

end for

3 损失函数及仿真验证

本次实验中，我们的任务是来预测用户是否会对某项物品感兴趣，因此可以看作是一个二分类问题，选用交叉熵来作为损失函数，交叉熵的数学形式如下：

$$l = -\sum_{ij} Y_{ij} \log \hat{Y}_{ij} + (1 - Y_{ij}) \log\left(1 - \hat{Y}_{ij}\right)$$

其在二维上的函数图像如图3.1所示：

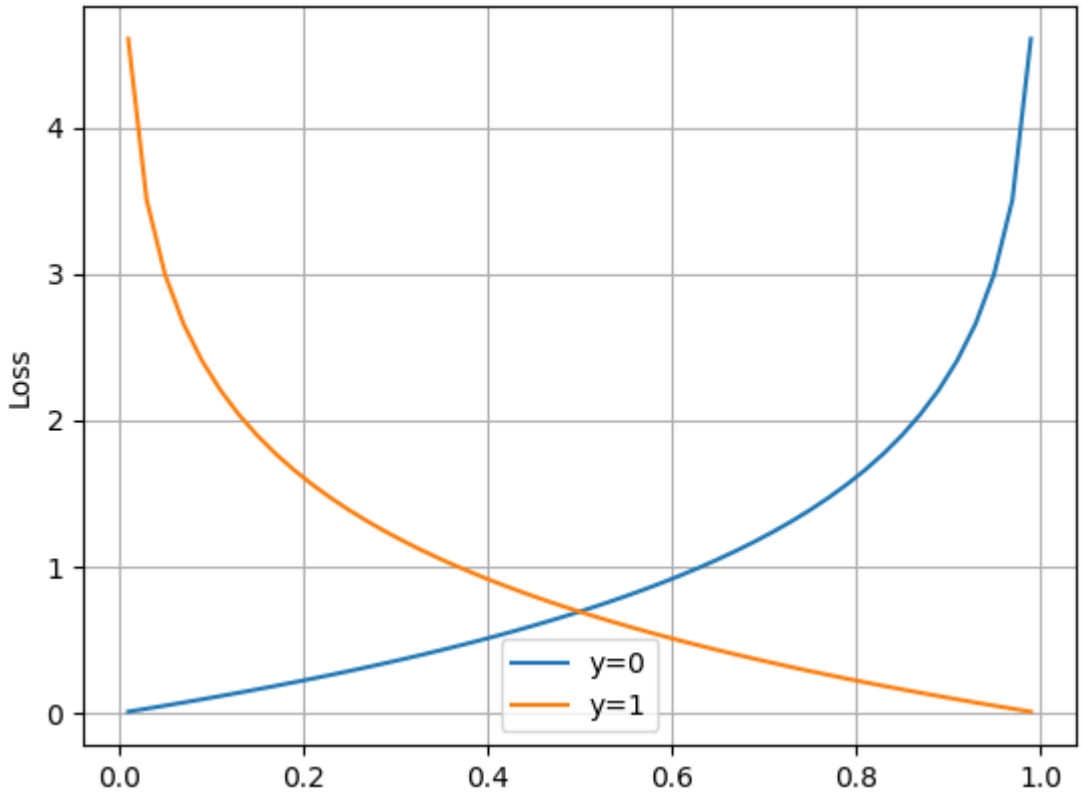


图3.1 交叉熵二维图像

这里，当实际标签为 1(y(i)=1) 时，函数的后半部分消失，而当实际标签是为 0(y(i=0)) 时，函数的前半部分消失。简言之，我们只是把对真实值类别的实际预测概率的对数相乘。

而在更高的维度下，通过模拟，我们可以得到交叉熵更直观的函数图像，如图3.2所示：

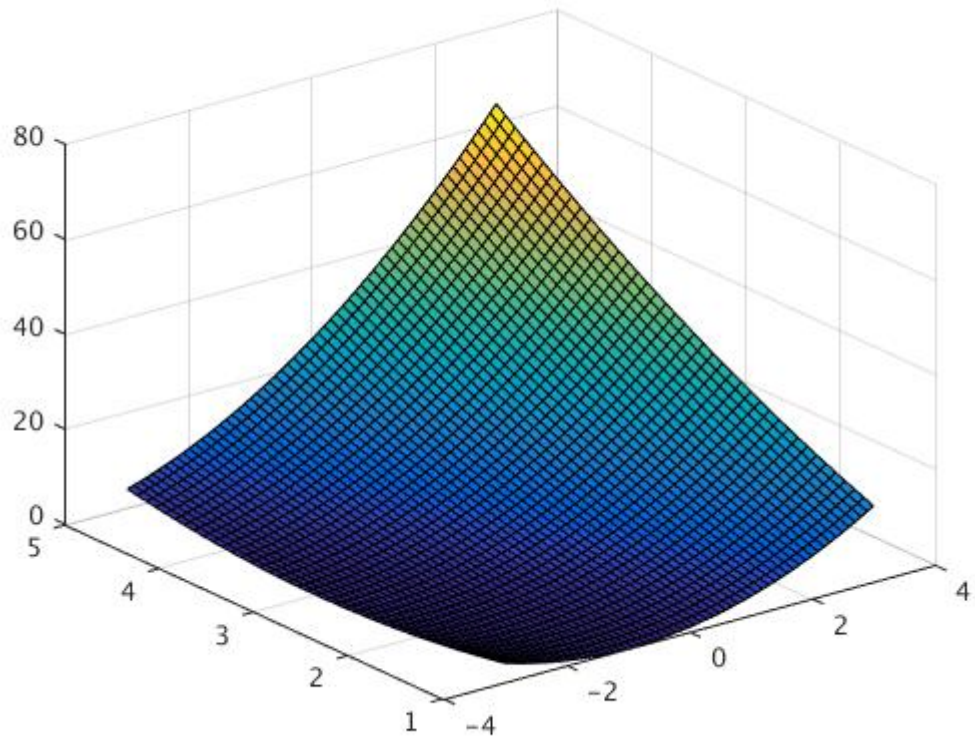
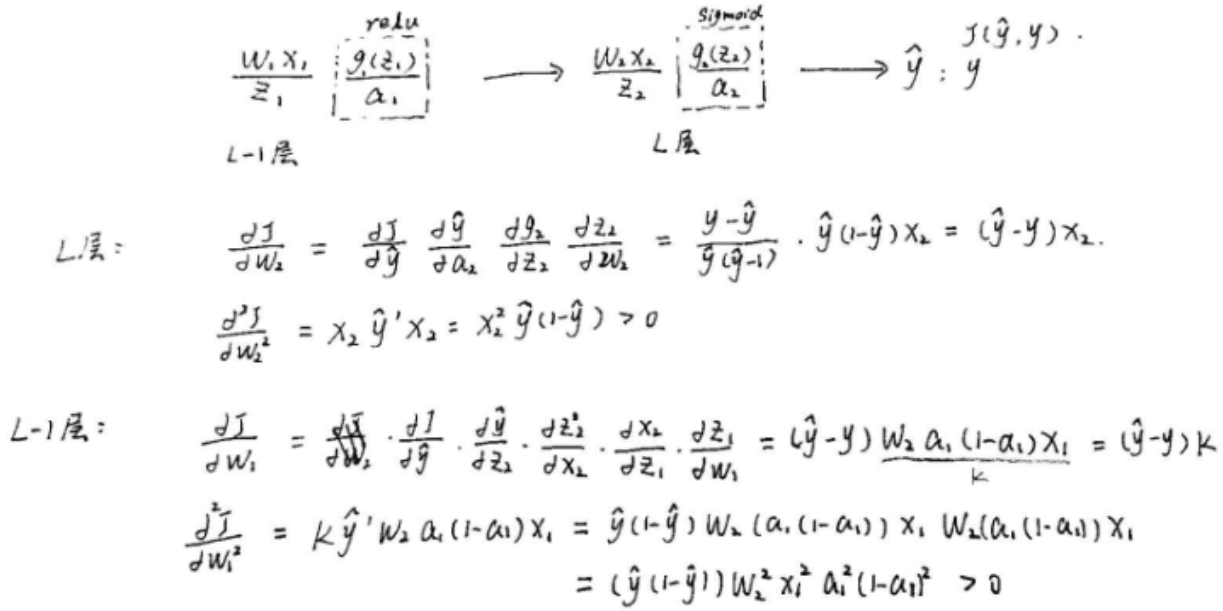


图3.2 交叉熵高维图像

神经网络的损失函数很多都是非凸的，但可以证明，在本实验中，如果代价函数是交叉熵，最后一层是sigmod函数，其他层是relu激活函数；那么此时的神经网络的损失函数是凸的。

证明如下：

假设函数f二阶可微，函数f是凸函数的充要条件是：其Hessian矩阵是半正定阵， 即$\nabla^2 f(x) \succeq 0$



因此，$J(\hat{y}, y)$ 是凸函数

4 实验结果及分析

我们将在MovieLens-1M和Pinterest两个真实公开的数据集中，分别测试之前介绍的三个模型，根据实验结果，分析MLFM以及FLFM相较于传统LFM模型在TopN推荐问题上的改进效果。与此同时，本文还将采用控制变量的方法，进行多组对照实验，探究模型的网络层数以及实验的负采样率等因素对最终TopN推荐效果的影响。

4.1 数据集介绍与预处理

数据集	用户数目	物品数目	评分数目	数据稀疏度
Movielens	6040	3952	1000209	95.81%
Pinterest	55187	9916	1500809	99.73%

本次实验是在MovieLens-1M[1]以及Pinterest[2]两个真实公开数据集上进行的。两个数据集的统计信息如上表中所示。

1. MovieLens数据集包含了6040个用户对3952部电影的评分，其评分在[1, 5]范围内，步长为1。而由于这是一个基于显示反馈的数据集，因此实验前，我们采用公式3.1来将其转化为隐式数据，即将所有用户有过评分的数据标记为1，将所有用户没有过评分的数据标记为0。

2. Pinterest是由[32]构建的用于作图像推荐的隐式反馈数据集。其中的交互信息表示的是用户是否将某个图片“钉”（pin）到了自己的画板上，反映的是用户对于不同图片的偏好。而由于原始数据过于庞大并且非常稀疏，为了更好地对我们的模型进行评估，我们采用了[22][28][31]等工作对于类似问题的处理方法：只保留评分

数在20及以上的用户和被评分达到5次及以上的物品。处理之后，数据集中共包含55187个用户对9916个物品总计1500809条评分记录。

4.2 实验设计

这一小节中，将讨论实验分析前的评价标准、数据集划分以及参数设置问题。

4.2.1 评价标准

本次TopN推荐的性能由命中率（hit ration，HR）以及归一化折损累积增益（Normalized discounted cumulative gain, NDCG）来进行衡量。

其中HR是一种召回率相关的指标，假设我们取推荐排名前K的物品作为推荐列表，那么HR就直观地衡量用户下一个交互的物品是否出现在推荐列表中。其公式为：

$$\mathrm{HR@K} = \frac{\mathrm{NumberofHits@}K}{|GT|}$$

其中|GT|是所有的测试集合，而分子则是每个用户前K个中属于测试集合的个数的总和。

而NDCG则是一种准确率相关的指标，衡量的是排序的质量。直观的来讲，命中物品在推荐列表中所处的位置越靠前，那么这一次推荐获得的分数也越高。其计算公式为：

$$\mathrm{NDCG@K} = Z_K \sum_{i=1}^{K} \frac{2^{\gamma_i} - 1}{\log_2(i+1)}$$

其中，r_i表示预测物品的相关性，当第i个位置的物品在测试结合中，那么$r_i = 1$，否则为0。而Z_k则是归一化系数，使得NDCG计算出来的结果在0到1之间。

4.2.2 数据集划分

本文采用留一法来评估top-N推荐的性能。具体做法是保留每个用户最近发生交互的物品作为测试集Y^+，而将剩余物品作为训练集。而由于为每个用户去将所有的物品进行排序是非常耗时的，于是在测试过程中，我们采用相关工作中常见的做法[28][33]随机抽取99个用户没有交互过的物品也加入到测试数据中，总计100个样本。接下来按照预测值来对列表中的100个物品进行排序，并截取前K个物品来计算上文提到的HR和NDCG两个指标，从而评估top-N推荐的具体性能。

4.2.3参数设置

在MLFM以及FLFM模型中，因为ReLU函数的稀疏激活以及收敛速度快的特性，我们选取$ReLU$函数来作为隐藏层的激活函数。而由于获得模型的预测结果是一个二分类问题，于是我们选取sigmoid函数来作为输出层的激活函数。在实际的训练过程中，模型则采用Adam方法来进行优化。而为了防止过拟合现象的发生，在实验过程中我们采用了dropout策略，以及L2正则化的方法来约束网络参数。其中模型的超参数设置如下表所示：

数据集	Batch size	Learning rate	Dtoupout rate
Movielens	256	0.001	0.5
Pinterest	512	0.002	0.5

4.3 实验结果分析

这里，为了方便起见，我们将可能会影响实验结果的相关因素定义如下:

网络层数：l

负采样率：negRatio

最顶层特征因子维度大小：f

推荐列表中物品数目：N

4.3.1 网络层数对实验结果的影响

在MLFM以及FLFM中，我们利用多个隐藏层来将用户和物品映射成为低维的隐语义表示。而为了确定模型最优的网络层数，我们将LFM，FLFM以及MLFM分别在MovieLens-1M数据集上，控制其他变量不变进行了实验，实验结果如下表所示：

实验模型（8Factors）	HR@10	NDCG@10
LFM	0.5884	0.3399
MLFM-1	0.5732	0.3207
MLFM-2	**0.6331**	**0.3637**
MLFM-3	0.6008	0.3365
MLFM-4	0.6189	0.3502
FLFM-1	0.6808	0.4009
FLFM-2	**0.6921**	**0.4085**
FLFM-3	0.6883	0.4028
FLFM-4	0.6869	0.4046

实验模型（16Factors）	HR@10	NDCG@10
LFM	0.6261	0.3738
MLFM-1	0.6434	0.3710
MLFM-2	**0.6742**	**0.3957**
MLFM-3	0.6621	0.3877
MLFM-4	0.6614	0.3864
FLFM-1	0.6965	0.4203
FLFM-2	**0.7050**	**0.4259**
FLFM-3	0.6942	0.4192
FLFM-4	0.6978	0.4187

为了消除其他因素的影响，在测试不同网络层数的时候，我们需要控制其他变量一致。这里我们取negRatio为4，N的大小为10，并分别测试f分别为8和16的情况下，不同网络层数l对于实验结果的影响。

表格中的MLFM-l以及FLFM-l分别代表l层的MLFM模型以及FLFM模型。其中我们将上一层的向量维度设置为下一层维度的2倍。例如，在f = 8的MLFM-3的模型中，模型的三层网络向量维度的大小分别为32，16，8。在没有特殊说明的情况下，下面的实验都将在这样的准则下进行。

从结果可以看到，MLFM的整体TopN推荐效果要明显好于传统的LFM，而融合了线性以及非线性特征的FLFM模型则又要好于MLFM模型。由此可见，在top-N推荐问题上，利用多层感知机学到的更深层非线性特征相比于传统LFM模型学习到的浅层特征，可以更好地反映出用户对物品的偏好以及物品的本质信息；而线性特征以及非线性特征的融合相比于单一特征则可以更充分地挖掘用户和物品的内在信息，从而取得更好地top-N推荐效果。

同时，从结果中可以看到，在f=8的情况下，2层的MLFM以及FLFM不管是在HR还是NDCG指标上都可以取得最好的推荐性能，而之后f=16的实验同样印证了这一点。值得注意的是，并不是更深的神经网络结构更有效。对于MLFM以及FLFM来说，3层和4层的网络结构在TopN推荐问题上并没有比2层的网络结构更有效。之后的实验结果分析中，在没有特殊说明的情况下，MLFM以及FLFM都将采用2层的网络结构。

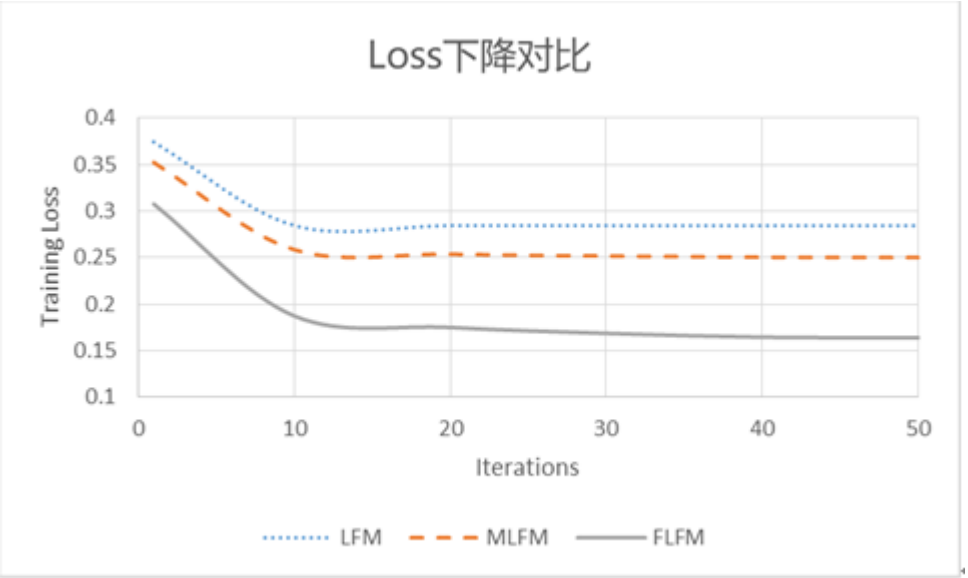


图4.2不同模型Loss下降对比

而在固定了N=10，f=16以及negRatio=4的前提下，图4.1则显示了LFM与2层的MLFM以及FLFM的Loss随着迭代次数下降的对比。这里可以看到，FLFM可以取得最小的训练误差，接下来是MLFM，最后是传统的LFM。这个结果也与上面三个模型在TopN推荐的效果相吻合，而这也表明，对于隐式数据来说，选取交叉熵作为损失函数来对参数进行优化是非常有效的。

4.3.2 负采样率对实验结果的影响

在训练采样的时候，要保证正样本与负样本数量的平衡，即负样本与正样本的比值是一个定值，也就是负采样率negRatio。本次实验中，我们在MovieLens-1M数据集中分别实验了MLFM与FLFM在负采样率为1，4，7，10，13情况下的TopN推荐性能，实验结果如下：

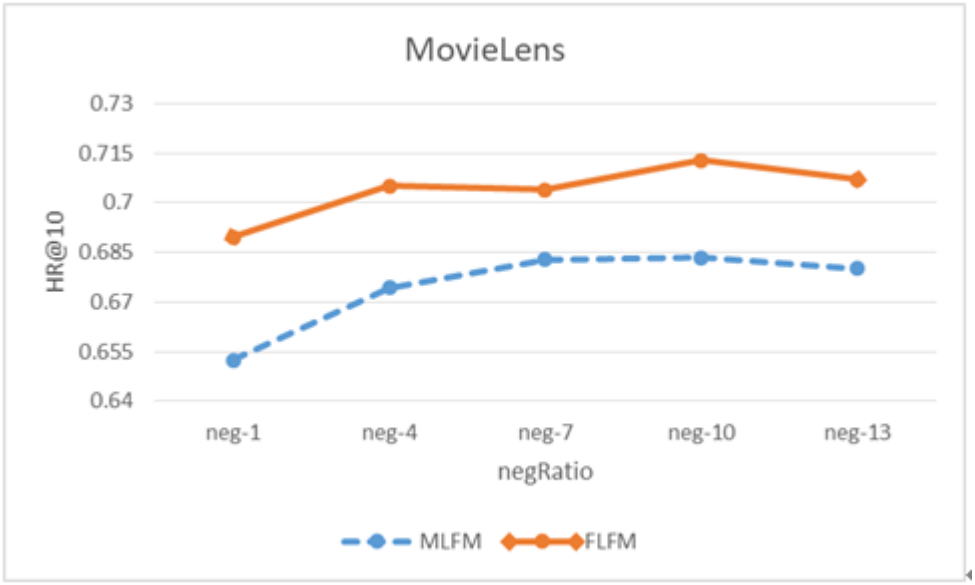


图4.2不同负采样率下HR@10对比

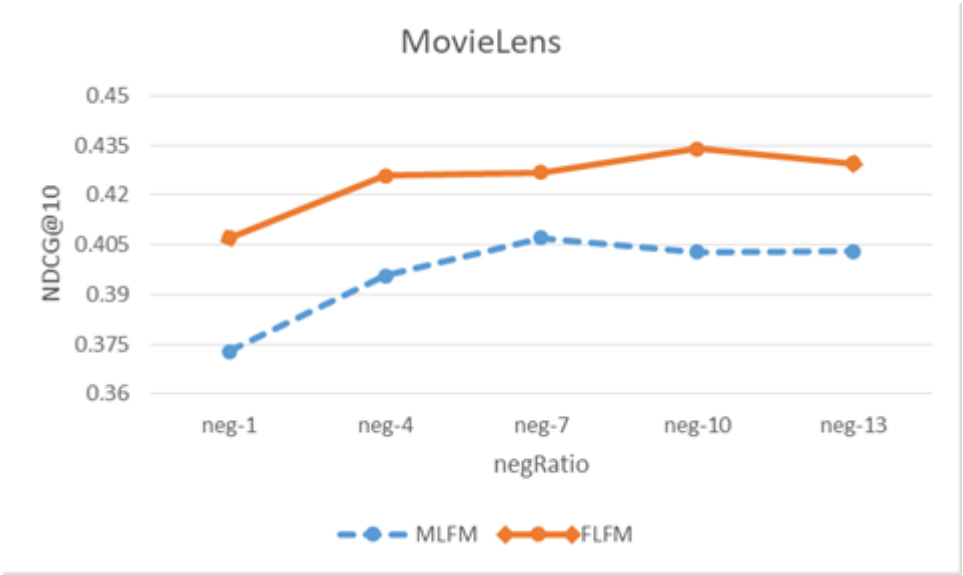


图4.3不同负采样率下NDCG@10对比

从实验结果可以看到，MLFM和FLFM均是在negRatio取到10的情况下取得最好的HR，而整体趋势也是在negRatio从1到10的变化过程中，HR逐渐增大，而在negRatio大于10的情况下，HR数值逐渐减小。而对于NDCG来说，情况也类似，只不过MLFM在negRatio为7的时候取得最大值，而FLFM则是在negRatio为10的时候取得最大值。但总体上FLFM的top-N推荐性能都要好于MLFM。

由实验结果分析可得，在negRatio数值小于10的时候，MLFM与FLFM的top-N推荐性能随负采样率的增大而增大；而在negRatio大于10的情况下，一方面训练的时间成本随之增加，而另一方面top-N的推荐性能也随之有轻幅度的滑落。因此在本次试验中，MLFM的最佳negRatio取值为7，而FLFM的最佳negRatio取值为10。

4.3.3 特征因子维度大小对实验结果的影响

网络最顶层的特征因子维度大小同样会影响实验的结果。同样，为了排除其他因素的影响，实验中固定negRatio值为4，N的数值为10，并分别在MovieLens以及Pinterest数据集上测试顶层特征因子维度大小从8到64情况下三个模型的top-N推荐性能，实验结果如下：

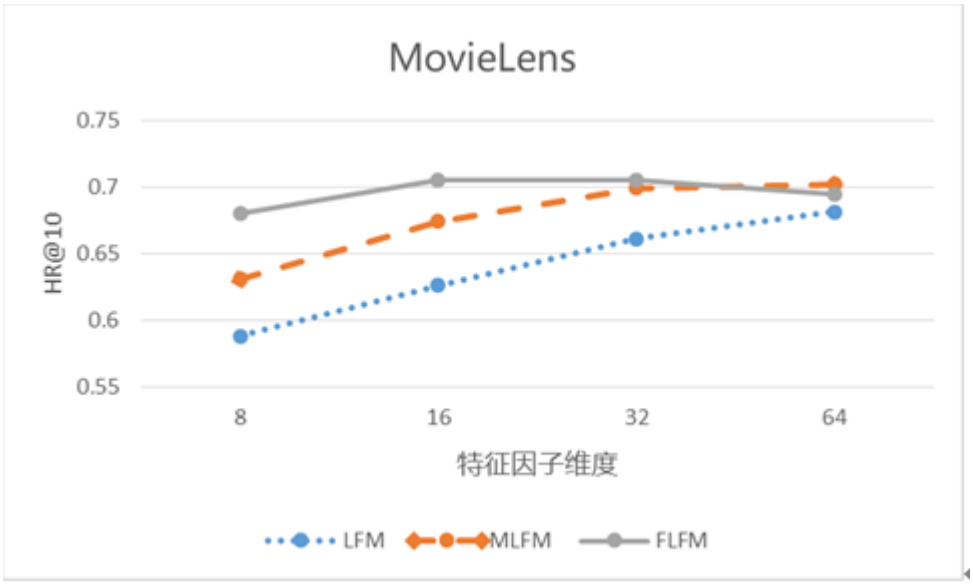


图4.4 MovieLens不同特征因子维度下HR@10对比

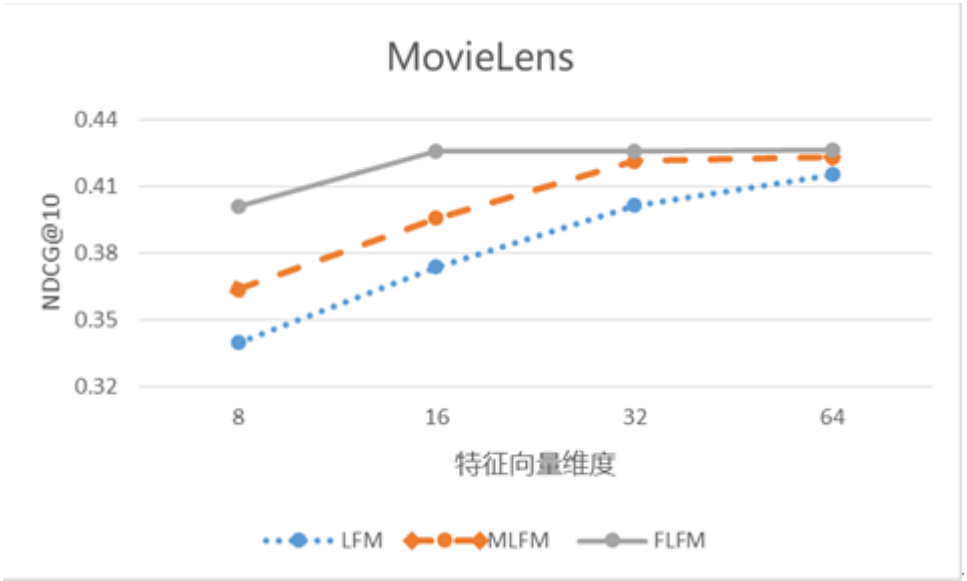


图4.5 MovieLens不同特征因子维度下NDCG@10对比

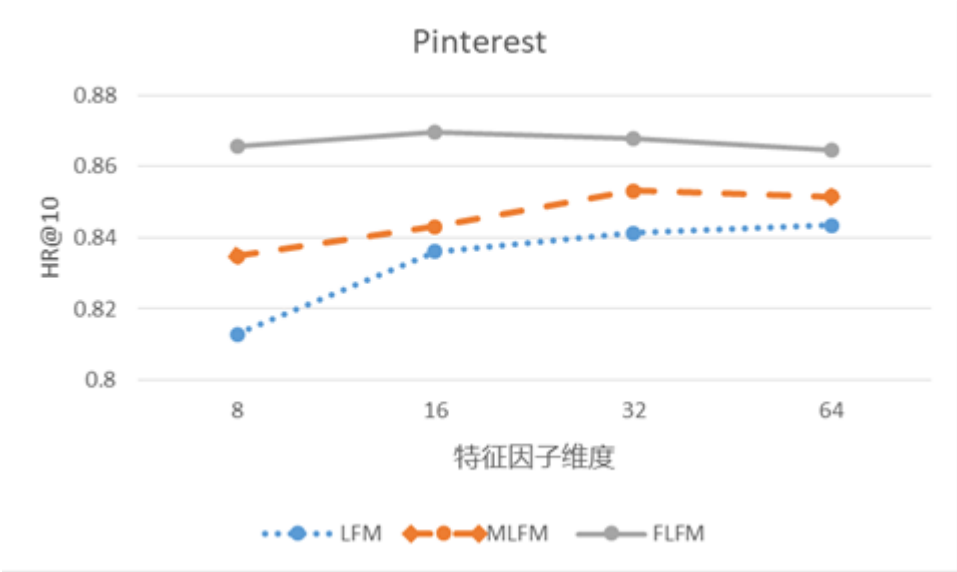


图4.6 Pinterest不同特征因子维度下HR@10对比

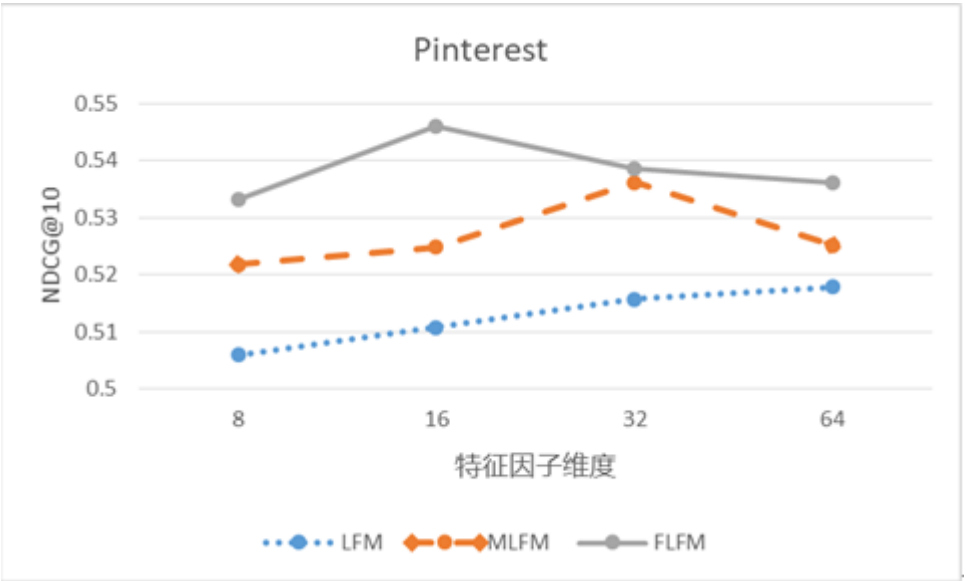


图4.7 Pinterest不同特征因子维度下NDCG@10对比

从实验结果中可以看出，LFM的top-N推荐性能随着特征因子维度大小f的增加而提升明显，而与之相比，MLFM在f达到32之后top-N推荐的性能略有下降，而FLFM与之类似，在f取到16的时候可以达到最好的top-N推荐性能。

而同时，可以看到LFM与FLFM在MovieLens数据集中在f达到64的时候已经非常接近甚至在某些指标上已经超过了FLFM的top-N推荐性能，而在更庞大更稀疏的Pinterest数据集上，FLFM的top-N推荐性能依旧明显好于MLFM以及LFM。由此可见，融合了线性特征以及非线性特征的FLFM在更庞大更稀疏的情况下依旧保持了非常好的推荐性能。

特征向量的维度总体上是来表征用户偏好以及商品内在特性的，从实验结果可以看出，传统的LFM往往需要更大的特征因子维度大小来表征用户偏好以及商品特性，而MLFM以及FLFM则能更有效地挖掘用户和物品的隐语义信息，使用较小的特征因子维度大小就可以得到不错的推荐性能。但同时，在特征因子维度过大的时候，特征因子往往会存在信息冗余的问题，导致过拟合现象的发生，这也解释了MLFM以及FLFM在f过大时推荐性能下降的原因。

4.3.4 推荐列表物品数目N对实验结果的影响

在实际推荐场景中，考虑到用户获取信息的精力有限，推荐系统在为用户推荐商品时需要控制推荐的列表大小。因此本文在控制negRatio为4，f为16的前提下，在MovieLens-1m以及Pinterest数据集中分别测试了推荐物品数目N取1到10的情况下，三种模型的推荐性能，如下所示：

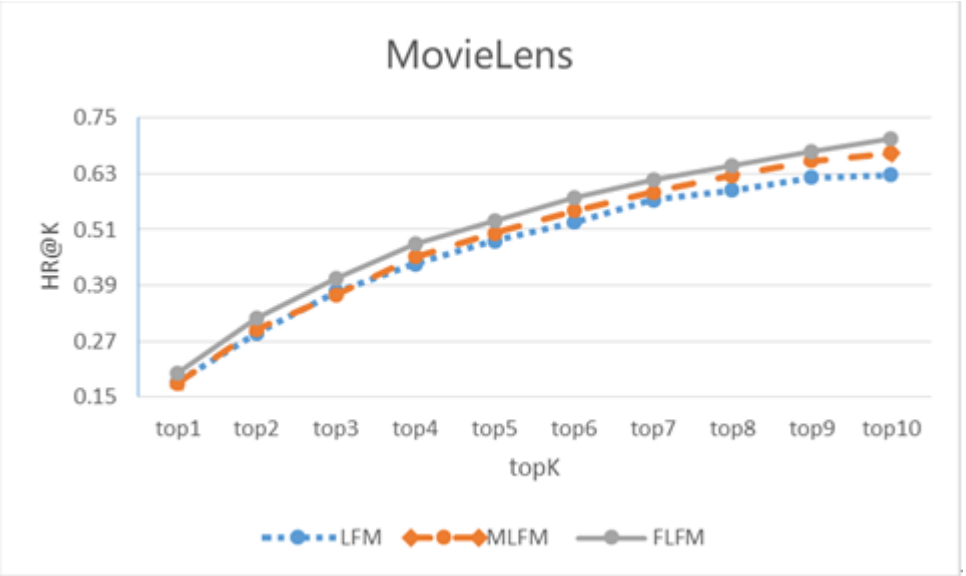


图4.7 MovieLens不同推荐数目下HR@10对比

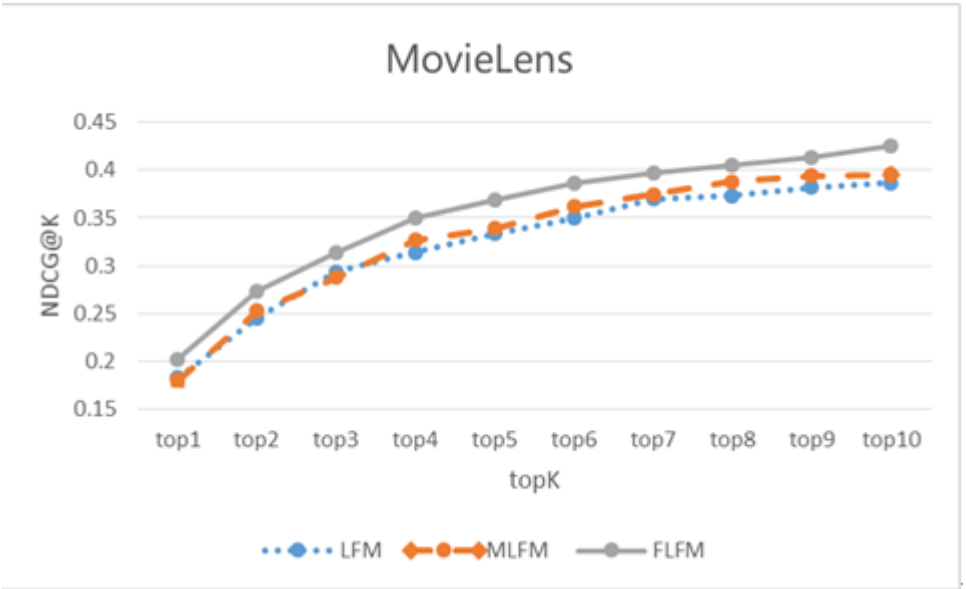


图4.8 MovieLens不同推荐数目下NDCG@10对比

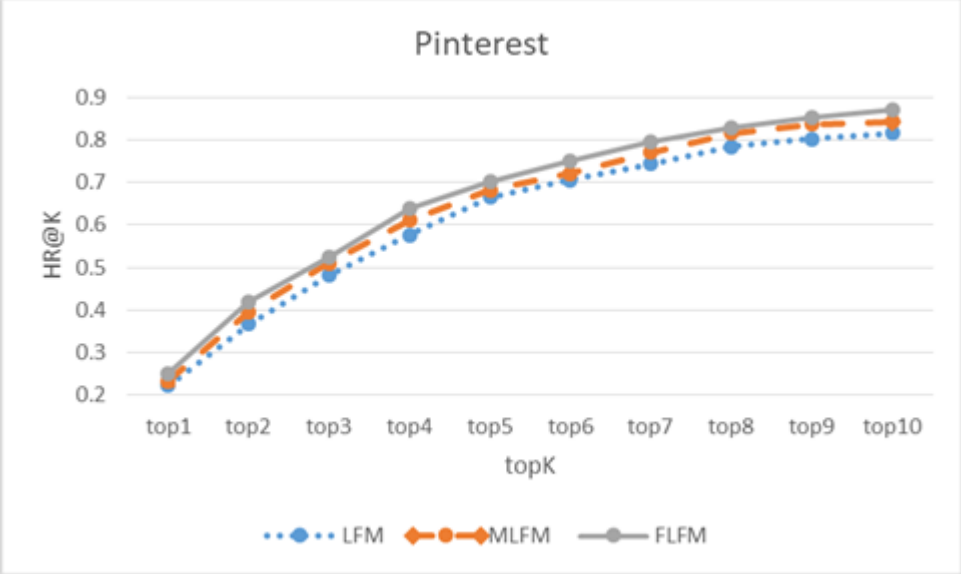


图4.9 Pinterest不同推荐数目下NDCG@10对比

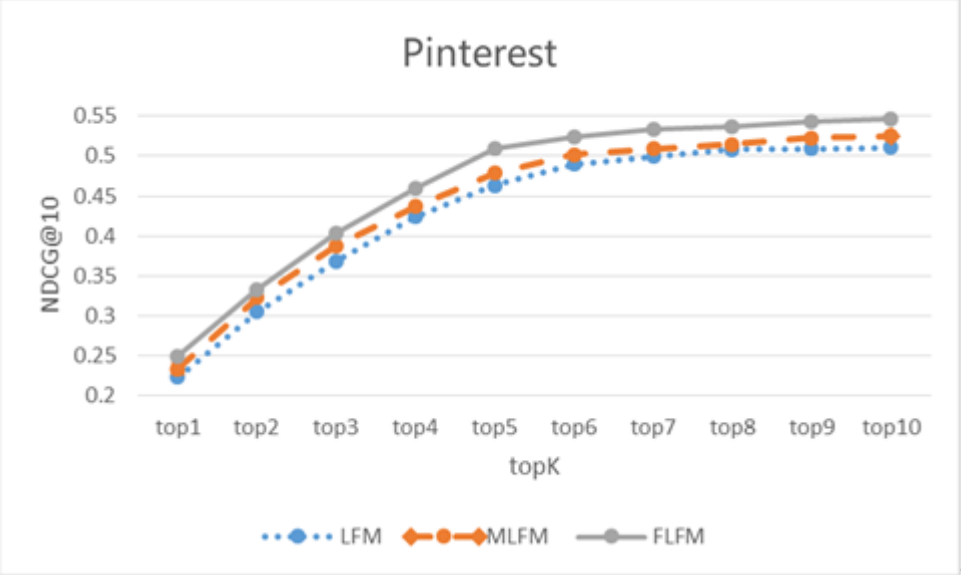


图4.10 Pinterest不同推荐书目下NDCG@10对比

---

[1] http://grouplens.org/datasets/movielens/1m/

[2] https://sites.google.com/site/xueatalphabeta/academic-projects

5 结论

从结果可以看到，MLFM的整体TopN推荐效果要明显好于传统的LFM，而融合了线性以及非线性特征的FLFM模型则又要好于MLFM模型。由此可见，在top-N推荐问题上，利用多层感知机学到的更深层非线性特征相比于传统LFM模型学习到的浅层特征，可以更好地反映出用户对物品的偏好以及物品的本质信息；而线性特征以及非线性特征的融合相比于单一特征则可以更充分地挖掘用户和物品的内在信息，从而取得更好地top-N推荐效果。

同时，从结果中可以看到，在f=8的情况下，2层的MLFM以及FLFM不管是在HR还是NDCG指标上都可以取得最好的推荐性能，而之后f=16的实验同样印证了这一点。值得注意的是，并不是更深的神经网络结构更有效。对于MLFM以及FLFM来说，3层和4层的网络结构在TopN推荐问题上并没有比2层的网络结构更有效。