豆瓣电影推荐

姓名	专业	学号
张玉洁	数学学院金融专硕	19210180014
詹晨	大数据学院应用统计	19210980066
林苗	数学学院应用统计	19210180051

1.绪论

1.1 背景与数据介绍

利用豆瓣电影的数据展示社交关系并进行个性化推荐。

数据分为用户信息表和电影信息表。其中用户信息表包括用户对部分电影的评分，以及描述用户喜好的关键词；电影信息表包括电影名、所属类型、评分、描述电影的关键词等信息。

1.2 基本思路与结构安排

先将用户之间的社交关系用可视化工具展示，然后分别用协同过滤以及机器学习、深度学习的方法进行个性化推荐。

第二节将介绍社交关系可视化。

第三节将介绍基于协同过滤的推荐：比较 4 个协同过滤模型，并将 4 个模型合并成一个模型，合并后的模型综合了 4 个模型各自的优点，最后的性能指标余弦相似度为 0.856，AUC 为 0.703，性能较好。

第四节将介绍基于机器学习与深度学习的推荐：尝试的模型有 GBDT+LR，FM 和 GBDT+DeepFM，最终结果为 GBDT+DeepFM 效果较好。

1.3 小组成员分工

林苗负责：用户之间社交关系的可视化与解读。

张玉洁负责：用户数据的清洗与特征提取，基于协同过滤模型的推荐。

詹晨负责：电影数据的清洗与特征提取，基于机器学习和深度学习的推荐。

2. 社交关系可视化

豆瓣数据包括电影数据和用户数据，电影数据包含四万多条影视剧的信息，清洗过后电影的特征主要有评分、导演、类型等标签，用户信息则包括每个人对感兴趣电影的评分和标签等。

表 2-1：movie 部分数据展示

number	id	title	rate	rating_people	directors	writers
1	3542816	我所追逐的是爱情	8.2	218	Archie Mayo	Maurice Hanline
2	26745334	游戏规则	4.3	3913	高希希	高希希
3	4014405	狂暴飞车	5.7	23389	帕特里克·卢西尔	Todd Farmer
4	1303976	白头神探 3	7.4	5095	彼得·西格尔	帕特·普罗夫

表 2-2：user 部分数据展示

user_id	name	rates
18	45123794	{u'6989755': u'1', u'4049665': u'4', u'6521838': u'4', u'4049667': u'4', u'3231622': u'5', u'2252778': u'3', u'2997052': u'4', u'1428365': u'5', }

连接两张表可以得出每个用户和其看过电影名称的评分表，我们对此表中的数据进行可视化。

2.1 同一导演的电影形成的社交关系

以个人风格明显的李安导演为例，选取他的所有电影及对电影进行打分的观众，连接用户和其评价过的电影，将用户对电影的评分作为边的权重，他们就可以通过共同评价电影而产生联系，这就形成了社交网络。

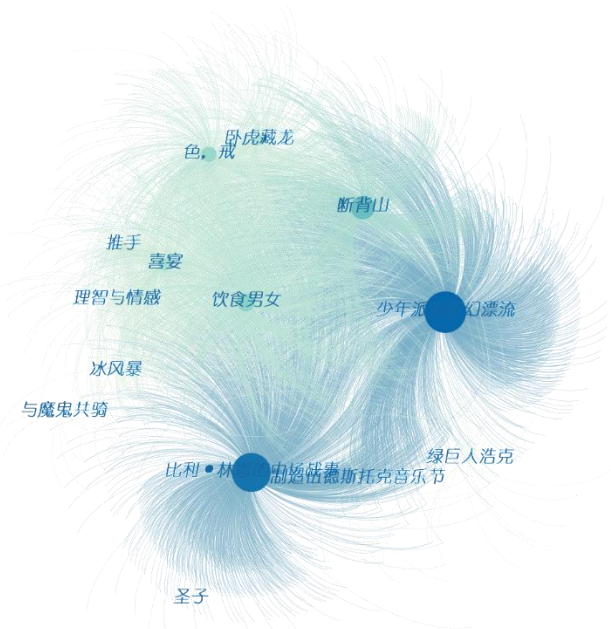


图 2-1：李安导演所有电影用户群社交图

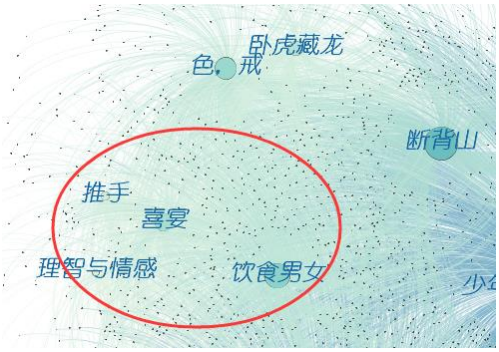


图 2-2：局部放大图

利用 gephi 输入节点表格和边表格作图，运行 Force Atlas 算法之后形成的布局如图 2-1 所示，蓝色节点表示电影,黑色节点代表观众，颜色越深说明入度越大评论人数越多，整张图的平均度在 4.329，平均加权度在 9.421。电影之间越接近表示电影的受众越相似，用户群交叉得多。可以看到图的网络直径较小，而且没有明显偏离整个中心的孤点出现。

而如图 2-2 所示，放大左上角，可以发现早年风格最为相似的父亲三部曲:《推手》、《喜宴》、《饮食男女》距离比其他电影更为接近，也说明在李安导演的“影迷群”中,还是有更为偏爱他早年风格的“怀旧党”们形成一个社区聚集在一块。

2.2 不同题材电影形成的社交网络

接下来探究对于风格题材导演都不相同的电影，社交网络图有何不同的表现。由于电影数据包含四万部不同的电影，打分用户少于 500 的冷门电影就有 25%左右，而用户信息则只包含豆瓣的部分用户评分信息，故有许多电影对应的打分用户过少，为保证数据的有效性，筛选出评论人数大于 10000 的热门电影，采取分层抽样的方法来展现风格迥异的电影之间的相似度。分层抽样的关键在于层内方差尽量小，层间方差尽量大，故对于不同题材如喜剧、科幻等 10 种题材进行分层，每层随机选择两部电影，共 20 部电影和其对应的观众做社交网络图。

表 2-3：按类型分层抽取的电影样本

层	编号	电影
喜剧	25870084	重返 20 岁
	24879858	分手大师
惊悚	1922273	电锯惊魂 4
	1895120	死寂
科幻	10453723	人类清除计划
	6873819	机器纪元
剧情	26279433	剩者为王
	26269510	陪安东尼度过漫长岁月
武侠	25919910	师父
	3726072	东邪西毒：终极版
战争	6879185	为奴十二年
	1295124	辛德勒的名单
家庭	1298694	恐龙
	10571509	如父如子
动作	5502697	夺宝联盟
	5364905	天机·富春山居图
动画	1304603	小飞象
	1295148	小鹿斑比
爱情	25845383	青春之旅
	25831308	露水红颜

如图 2-3 所示，整张图的网络直径较大，并且小世界效应也比图 2-1 更为明显。图 2-3 的平均度为 3.927，加权度 7.027，均小于图 1，也可以看出李安导演相对于影片的平均水平来说,电影评论人数更多,平均评分也即口碑更为优秀。

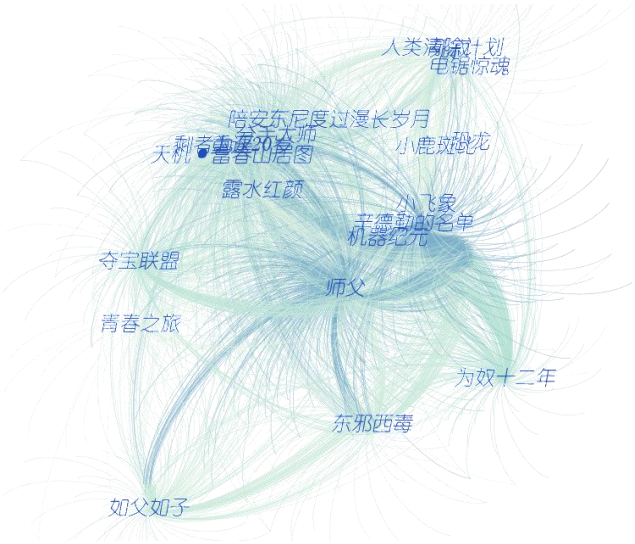


图 2-3：不同类型电影用户社交网络图

具体来看图 2-3，左上角的陪安东尼度过漫长岁月和剩者为王、露水红颜等商业片明显与辛德勒的名单、为奴十二年这类经典电影画出界限，电影受众群重叠较少；而右上角科幻惊悚片如人类清除计划、电锯惊魂等更像是小众情怀。从地区来说，相同地区的电影也表现出更高的相似性，唯一一部日本电影如父如子与图中心相距较远，韩国电影夺宝联盟和青春之旅的用户群体形成一个社区，左上角最为密集的商业剧情片则都是中国电影。不同的社区之间用户对于电影的偏好明显不同。

3. 基于协同过滤的推荐

3.1 模型介绍

3.1.1 基于物品的协同过滤

基于物品的协同过滤算法的核心：给用户推荐那些与他们之前喜欢的电影相似的电影。计算电影相似度运用了以下几种方法：

相似度计算

（1） 基于共同评价该电影的用户列表计算（带有对热门电影的惩罚项）

$$w_{ij} = \frac{|N(i) \cap N(j)|}{|N(i)|^{\alpha} * |N(j)|^{1-\alpha}}$$

其中,分母中$N(i)$表示评价电影 i 的用户集合，$N(j)$表示评价电影 j 的用户集合,分子$N(i) \cap N(j)$表示同时评价电影 i 和电影 j 的用户集合。|A|表示集合 A 的模。

系数α表示对热门电影 i 的惩罚,对于热门电影，分子$N(i) \cap N(j)$的增长速度往往高于$N(i)$，这就会使得电影 i 和很多其他电影的相似度都偏高，使得推荐结果过于热门，从而使得个性化感知下降，所以我们需要添加惩罚项α，当$\alpha \in (0,0.5)$时，$N(i)/N(j)$越小,惩罚得越厉害，从而会使热门电影的相关性分数下降。

这个公式的核心是计算同时评价这两部电影的人数比例，当同时评价过这两部电影的人数越多时，我们认为这两部电影的相似度越高。

（2）基于余弦相似度计算

将电影的用户评分数据与电影的信息数据合并形成电影的特征，计算余弦相似度。公式如下：

$$w_{ij} = \cos\theta = \frac{N_i \bullet N_j}{\|N_i\| \|N_j\|} = \frac{\sum_{k=1}^{len}(n_{ki} \times n_{kj})}{\sqrt{\sum_{k=1}^{len} n_{ki}^2} \times \sqrt{\sum_{k=1}^{len} n_{kj}^2}}$$

其中len表示电影特征的长度，n_{ki}表示第 i 部电影的特征向量N_i的第 k 个分量。

这个公式的核心是我们认为用户对两部电影的评分越相似，比如对 i 打低分的也会对 j 打低分，则两部电影越相似。

预测评分

得到物品的相似度之后,我们预测用户 u 对一个电影 i 的打分：

$$p_{uj} = \sum_{i \in N(u) \cap S(j,k)} w_{ji} score_{ui}$$

其中$S(j,k)$是物品 j 最相似的 Top-k 个物品的集合,$N(u)$为用户 u 评分的电影的集合,$score_{ui}$为用户 u 对电影 i 的评分。

3.1.2 基于用户的协同过滤

基于用户的协同过滤原理和基于物品的协同过滤相似。所不同的是，基于物品的协同过滤的原理是用户 U 购买了 A 物品，推荐给用户 U 和 A 相似的物品 B、C、D。而基于用户的协同过滤，是先计算用户 U 与其他用户的相似度，然后取和 U 最相似的几个用户，把他们购买过的物品推荐给用户 U。

与基于用户的协同过滤原理类似，我们可以用以下公式计算相似度：

（1）基于共同评价的电影的计算

$$w_{ab} = \frac{|N(a) \cap N(b)|}{\sqrt{|N(a)| * |N(b)|}}$$

其中丨$N(a)$丨表示用户 a 评价的电影的数量，丨$N(b)$丨表示用户 b 评价的电影的数量,$N(a) \cap N(b)$表示用户 a 与用户 b 评价相同电影的数量。

这个公式的核心是两位用户共同评价的电影越多，我们就认为他们越相似。

（2）余弦相似度

与电影间相似度计算同理，我们可以根据用户对电影的评分数据计算余弦相似度。

预测评分

得到用户的相似度之后，我们预测用户 u 对物品 i 的打分

$$p_{ui} = \sum_{v \in N(u) \cap S(u,k)} w_{vu} \mathrm{score}_{vi}$$

其中$S(u,k)$是与用户 u 相似的 Top-k 个用户的集合，$N(u)$为用户 u 评分过的电影的集合,$score_{vi}$为用户 v 对电影 i 的评分，$w_{u,v}$为用户 u 与用户 v 的相似度。

3.2 模型实现与结果分析

3.2.1.数据预处理

由于运算量过多，将原电影评分数据删减至最热门的 3000 部电影,热门程度根据对电影的评分用户数计算，最后我们得到 5000 个用户对 3000 部电影的评分；使用 4 折交叉验证法,将数据分为训练集与测试集。

3.2.2.对模型的修正与调节

（1）未定参数 k 与$\boldsymbol{\alpha}$

k 的选取

在预测用户对电影的评分时，我们分别在基于物品的与基于用户的协同过滤中召回了最相似的 k 个物品的集合与最相似的 k 个用户的集合，这个 k 为未定参数。过大的 k，会召回很多相关性不强的物品，导致准确度下降；过小的 K 会使得召回的物品过少，使得准确度也不高。在这里，我们尝试不同的 K 值对比算法的性能，以便选择最佳的 k 值。

对于 K 的选取，如图 3-1 是训练集中用户评分数的概率分布图，横坐标表示用户评价电影的数量，纵坐标表示用户数量。从表中我们可以看到，在评价电影数量为 0-100 部时，用户占了绝大多数，为$2738/5000 = 54.76\%$，故在对召回数 K 进行调参时，我们选取 $k = 10$，20，30,…,90,100。

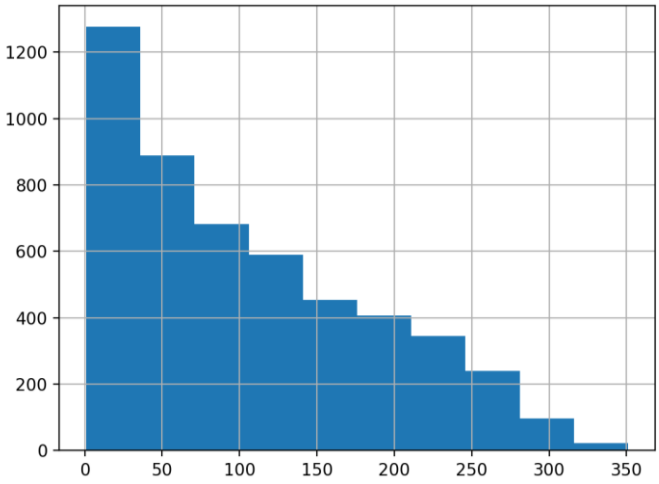


图 3-1：训练集中用户评分数的概率分布图

α的选取

由上提及，因为$\alpha \in (0,0.5)$, 所以我们取$\alpha = 0.3,0.4,0.5$。

（2）同时，考虑到在基于物品的协同过滤模型计算评分时，用户不一定对于召回的 K 个物品都有评分，因此，将模型修正为两种情况：

- 仍使用召回的最相似的 K 个物品，若用户对其中某些物品没有评分，则记评分为 0。即原模型的

$$p_{uj} = \sum_{i \in N(u) \cap S(j,k)} w_{ji} score_{ui}$$

其中$S(j,k)$是电影 j 最相似的 k 个电影的集合，$N(u)$为用户 u 评分的电影的集合，$score_{ui}$为用户 u 对电影 i 的评分。

- 在用户评价过的物品中，找与评分物品最相似的 k 个物品。即

$$p_{ui} = \sum_{N(u) \cap S(j,n)} w_{ji} score_{ui}$$

其中$S(j,n)$是电影 j 最相似的 n 个电影的集合，$N(u)$为用户 u 评分的电影的集合，且满足$|S(j,n) \cap N(u)| = k$, $score_{ui}$为用户 u 对 i 的评分。但在模型训练中，我们添加约束条件$n \leq 2k$。

我们称上面两种情况分别为召回计数方式 Method1，召回计数方式 Method2。

对 k 个最相似的用户我们也有类似的处理方法。

(3)模型总结

基于对物品的协同过滤和基于对用户的协同过滤，以及相似度计量的不同，我们总共可以得到 4 个浅层模型，我们需要对 4 个浅层模型都进行调参。

表 3-1：浅层模型

模型定参				
	模型	**相似度度量**	**参数调节**	**召回计数方式**
基于物品的协同过滤	**模型 11**	基于共同评价的用户的计算 $w_{ij} = \frac{\lvert N(i) \cap N(j) \rvert}{\lvert N(i) \rvert^{\alpha} * \lvert N(j) \rvert^{1-\alpha}}$	α，k	Method1 or Method2
	模型 12	基于余弦相似度	k	Method1 or Method2
基于用户的协同过滤	**模型 21**	基于共同评价的电影的计算 $w_{ab} = \frac{\lvert N(a) \cap N(b) \rvert}{\sqrt{\lvert N(a) \rvert * \lvert N(b) \rvert}}$	k	Method1 or Method2
	模型 22	基于余弦相似度	k	Method1 or Method2

3.2.3 评价指标

（1）二分类评价指标 AUC

把打分模型转化为二分类问题。基于评分数据,对于某一用户，将预测评分最高的十个物品判为 1,剩下的判为 0。在测试集中，将该用户评分最高的十个物品判为 1,剩下的判为 0。对此二分类结果计算 AUC。

（2）评分评价指标余弦相似度

由于协同过滤模型预测了用户的评分数据，对于某一用户，我们希望测试集中该用户评分低的物品预测分数也低，该用户评分高的物品预测分数也高，因此将用户的评分与测试集中的评分计算余弦相似度。

3.3.3 模型预测结果

其中 B1COS,B1AUC 分别表示模型召回计数方式 Method1 的余弦相似度与 AUC，B2COS，B2AUC 分别表示模型召回计数方式 Method2 的余弦相似度与 AUC。

表 3-2：模型预测结果

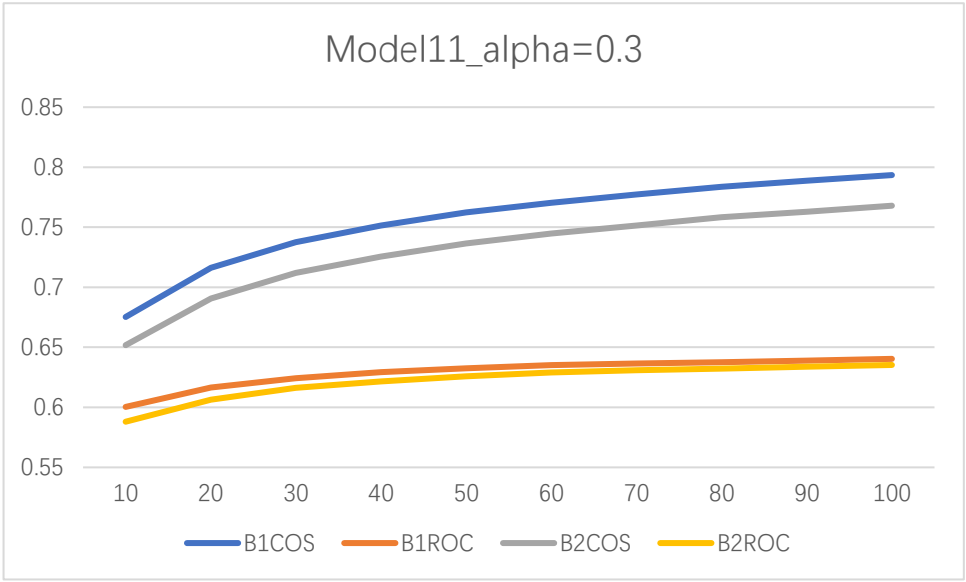
model1_alpha=0.3				
k	**B1COS**	**B1AUC**	**B2COS**	**B2AUC**
10	0.67526947	0.60036018	0.65177467	0.58802979
20	0.71620675	0.61645513	0.69059403	0.60635418
30	0.73765401	0.62419794	0.71197101	0.61634806
40	0.75142031	0.62918821	0.7257022	0.62164083

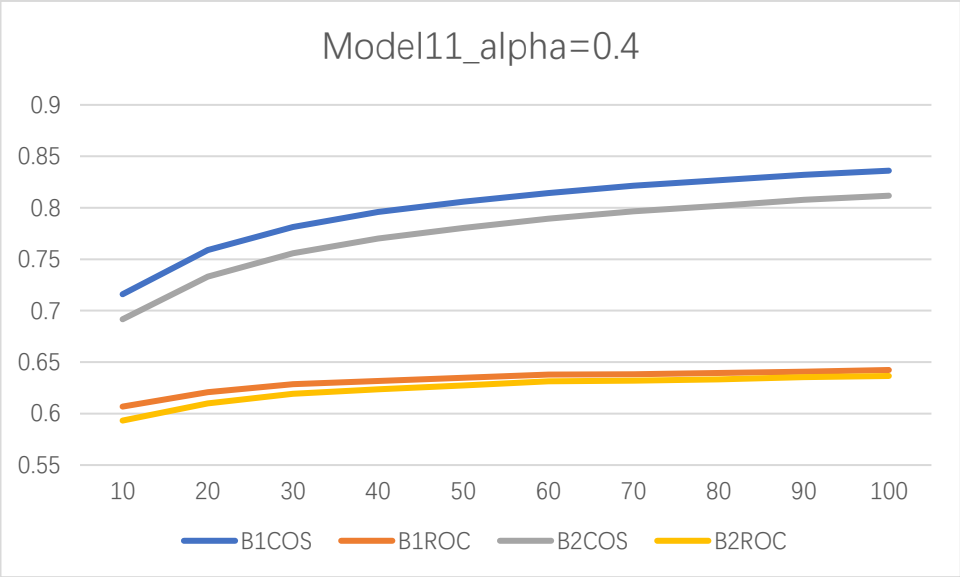
50	0.76247596	0.63248484	0.73655272	0.62579799
60	0.77039529	0.6352191	0.74475553	0.62909143
70	0.77727824	0.63657843	0.75146534	0.63094373
80	0.78365656	0.63770401	0.75826015	0.63235752
90	0.78867095	0.63902042	0.76279381	0.63394223
100	0.7934411	0.64037503	0.76795386	0.63530581
model1_alpha=0.4				
k	B1COS	B1AUC	B2COS	B2AUC
10	0.71598398	0.60682704	0.69167565	0.59324726
20	0.75894807	0.62095805	0.73300852	0.60995767
30	0.78149064	0.62865835	0.7557514	0.61936821
40	0.79588537	0.63165409	0.77024992	0.62370937
50	0.80587643	0.63470179	0.78048759	0.62745538
60	0.81432822	0.63785191	0.7893242	0.63132379
70	0.82149429	0.63839312	0.79664378	0.63191421
80	0.82668637	0.63951939	0.80199807	0.63334796
90	0.83190266	0.6408131	0.80789067	0.63548966
100	0.83605532	0.64238423	0.81173695	0.63657137
model1_alpha=0.5				
k	B1COS	B1AUC	B2COS	B2AUC
10	0.75608571	0.61180594	0.73156488	0.59805987
20	0.79942795	0.62375397	0.77424863	0.61321282
30	0.82124547	0.63021099	0.79565246	0.6205918
40	0.83532777	0.63384973	0.8102177	0.62573848
50	0.84595699	0.63625376	0.82113198	0.62892
60	0.85357735	0.63878506	0.82937707	0.63237731
70	0.85914526	0.64017894	0.83455141	0.63353895
80	0.86492661	0.64153615	0.84128113	0.63521667
90	0.86924325	0.64297975	0.84534053	0.63698159
100	0.87315038	0.64336696	0.84959496	0.63739995
model2				
k	B1COS	B1AUC	B2COS	B2AUC
10	0.75133	0.61253269	0.37919866	0.60132279
20	0.79934645	0.62650357	0.41542722	0.62697665
30	0.82100612	0.63302547	0.43575958	0.63966906
40	0.83575231	0.63673351	0.45189288	0.64918247
50	0.84617602	0.63867537	0.46397473	0.65485969
60	0.85380718	0.64156305	0.47402858	0.66016841
70	0.8604347	0.64201355	0.48202894	0.66345025
80	0.86555818	0.64432027	0.48918337	0.66699207
90	0.87044611	0.64515726	0.49662592	0.66945571
100	0.87380605	0.64580439	0.50213579	0.67208855
model21				

k	B1COS	B1AUC	B2COS	B2AUC
10	0.69764526	0.63617705	0.74791487	0.66438843
20	0.73548412	0.66207194	0.76875164	0.68242221
30	0.75015425	0.67300667	0.77585025	0.68922209
40	0.75919159	0.68088944	0.78039642	0.69442287
50	0.76451026	0.68514245	0.78334721	0.69671736
60	0.76788167	0.68787476	0.78460689	0.69788502
70	0.77060647	0.69085217	0.78601733	0.70038888
80	0.7732257	0.69338821	0.7866938	0.70223267
90	0.77535184	0.69532718	0.78800168	0.70292157
100	0.77669995	0.69766669	0.788639	0.70454235
model22				
k	B1COS	B1AUC	B2COS	B2AUC
10	0.75996672	0.66856836	0.80460767	0.69243918
20	0.78670717	0.6937746	0.81597232	0.70902257
30	0.79601433	0.6986456	0.8189447	0.70980014
40	0.80019158	0.70587923	0.81960314	0.71474252
50	0.80304034	0.70873822	0.82005562	0.71639484
60	0.80450597	0.70849059	0.81952373	0.71414667
70	0.80580582	0.70995124	0.81960791	0.71457854
80	0.8066292	0.71270135	0.81926158	0.71712777
90	0.80750693	0.71192914	0.81907288	0.7162498
100	0.80741232	0.71119435	0.81816915	0.71466077

3.3.4 模型预测结果分析与定参结果

（1）Model11





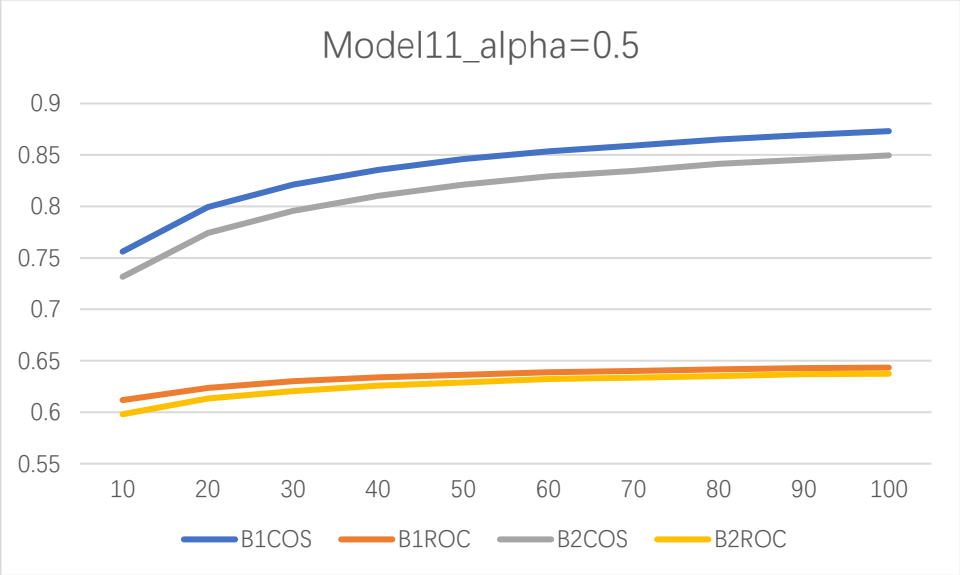


图 3-2：模型预测结果图

从图中可以看到，model11 的 2 种召回计数的 AUC 和 ROC 都随着 k 的增大而增大，且第一种召回计数方式的准确度要明显大于第二种召回方式的准确度，因此对 model11，我们采用第一种召回方式，且取 k=100。

但$\alpha = 0.5$时，model11 的准确度要高于$\alpha = 0.3,0.4$时，说明豆瓣电影评分用户对热门电影具有一定的偏好性，所以我们对 model11,我们选取$\alpha = 0.5$。

表 3-3：model11 调参结果表

model11_k=100				
alpha	B1COS	B1AUC	B2COS	B2AUC
0.3	0.7934411	0.64037503	0.76795386	0.63530581
0.4	0.83605532	0.64238423	0.81173695	0.63657137
0.5	0.87315038	0.64336696	0.84959496	0.63739995

（2）Model12

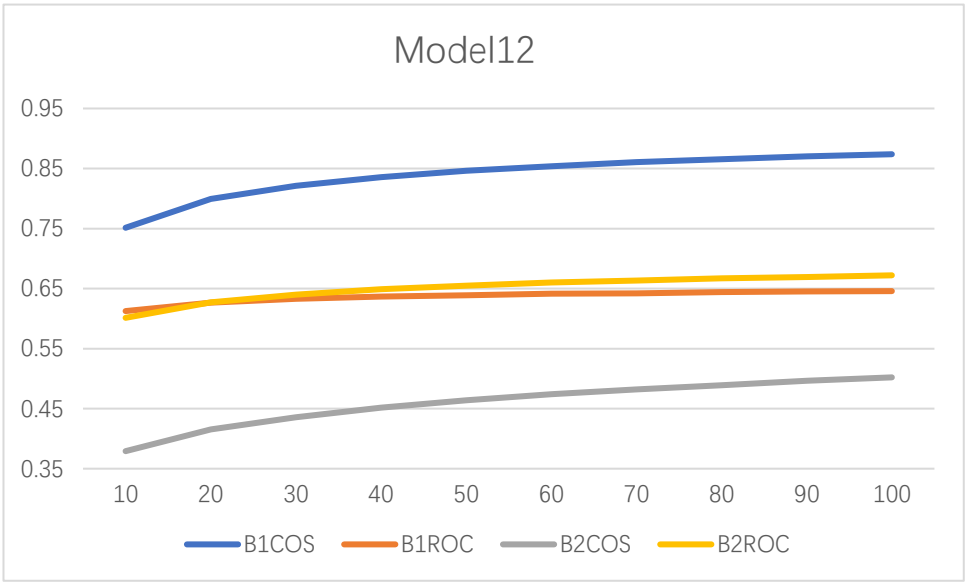


图 3-3：模型 model12 预测结果图

对于模型 12，从图 3-3 中可以看到，Model12 的 2 种召回计数的 AUC 和 ROC 都随着 k 的增大而增大，且第一种召回计数方式的 COS 准确度要明显大于第二种召回方式的 COS 准确度，而第一种召回计数方式的 AUC 准确度要略低于第二种召回方式的 AUC 准确度因此对 model12，我们采用第一种召回方式，且取 k=100。

（2） Model21

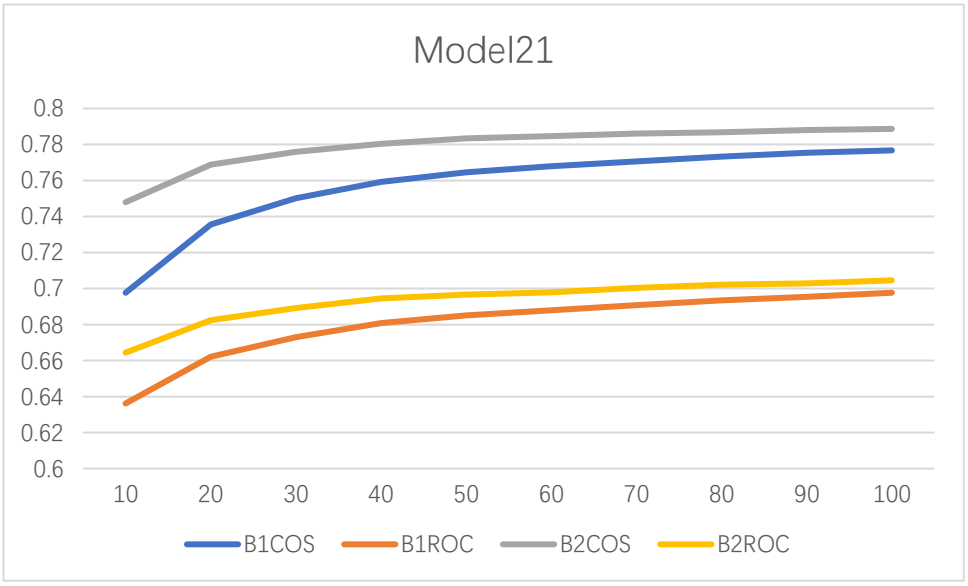


图 3-4：模型 model21 预测结果图

对于 Model21，从图中可以看到，Model21 的 2 种召回计数的 AUC 和 ROC 都随着 k 的增大而增大，且第二种召回计数方式的准确度要大于第一种召回方式的准确度，因此对 model21，我们采用第二种召回方式，且取 k=100。

（3） Model22

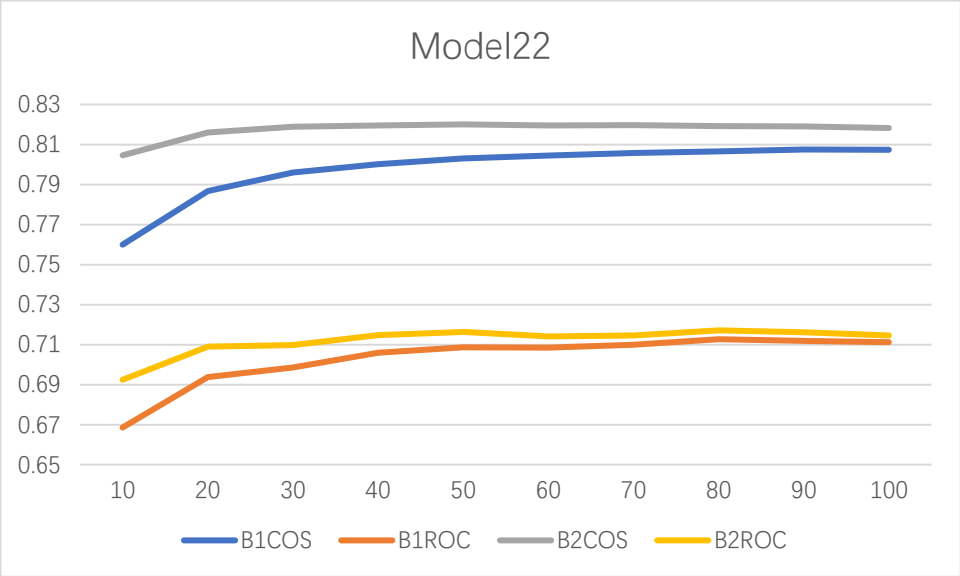


图 3-5：模型 model22 预测结果图

对于 Model22，从图中可以看到，第二种召回计数方式的准确度要明显大于第一种召回计数方式的准确度，且在 k=50 时，B2COS 达到最高值，而 B1AUC 总体来说随着 k 的变化不大。因此对 model22，我们采用第二种召回方式，且取 k=50。

表 3-4：调参结果表

模型定参				
	模型	**相似度度量**	**参数调节**	**召回计数方式**
基于物品的协同过滤	**模型 11**	**基于共同评价的用户的计算** $w_{ij} = \frac{\lvert N(i) \cap N(j) \rvert}{\lvert N(i) \rvert^{\alpha} * \lvert N(j) \rvert^{1-\alpha}}$	**α=0.5 K=100**	**Method1**
	模型 12	**基于余弦相似度**	**K=100**	**Method1**
基于用户的协同过滤	**模型 21**	**基于共同评价的电影的计算** $w_{ab} = \frac{\lvert N(a) \cap N(b) \rvert}{\sqrt{\lvert N(a) \rvert * \lvert N(b) \rvert}}$	**K=100**	**Method2**
	模型 22	**基于余弦相似度**	**K=50**	**Method2**

3.3.5 模型合并

将定参后四个浅层模型的评分数据归一后相加，作为最终评分结果，并对合并后模型进行性能评价。

表 3-5：最终评分结果

	AUC	余弦相似度
模型 11	0.643367	0.87315
模型 12	0.645804	0.873806
模型 21	0.704542	0.788639
模型 22	0.714661	0.818169
合并后模型	0.703867	0.855246

可以看到，合并后的模型均衡了基于用户的协同过滤在二分类问题 AUC 上的优点和基于物品的协同过滤在余弦相似度上的优点。对于二分类问题的 AUC 指标，我们总可以用 0.5 作为衡量其性能的标准；而对于余弦相似度，我们计算 5000 名用户中任意两名用户对 3000 部电影的评分的余弦相似度的均值为 0.100384，以这个为基准，可以看出我们的浅层模型的余弦相似度准确度较高。

4. 基于机器学习与深度学习的推荐

4.1 数据预处理

数据预处理包含三个部分，第一部分是电影的特征处理，第二部分是用户信息的特征处理，第三部分是表的连接。

4.1.1 电影的特征处理

电影的描述关键词为表中的“tags”字段，用“#”分隔。筛选出频次大于10的描述词汇，然后用Tf-idf编码；电影类型为表中的“type”字段，用“/”分隔的词汇，发现有些电影类别是同一个类别，只是表达方式不一样，比如“惊悚”和“惊栗”，人工定义转换词典，把同一种电影类型用同一种表达方式表示，然后用one-hot编码成向量；演员特征考虑到电影的主演往往会影响影迷是否会观看这部电影，而配角往往不会产生影响，并且出现过的演员很多，如果编码成向量维度会很大，所以取每个电影的演员列表的前3名，用one-hot编码，作为电影的有关于演员方面的特征；其他特征，比如评论人数、评分和评论人数都为数值型，无需做其他处理。

4.1.2 用户的特征处理

用户的特征用用户表中的“tags”字段中关键词用Tf-idf编码成向量。

4.1.3 表的连接

用户表中包含用户对于某些电影的打分，最终整理成<用户信息，电影信息，打分>。每一个样本为用户的表示向量和电影的表示向量与用户为该电影的打分。后续模型中把分数大于3的视作用户喜欢这部电影，记为1，应该给用户推荐该电影。

4.2 GBDT+LR

4.2.1 模型介绍

把用户打分高的电影看作我们要推荐的电影，把打分低的电影看作不推荐的电影，那么电影推荐问题可以看作一个二分类问题。那么可以用逻辑回归等模型解决，但是缺点是逻辑回归是线性模型，特征进行线性组合，学习能力有限，往往效果不好。

而如果想要考虑多个特征的组合关系，可以用FM（Factorization Machine）模型，FM模型见下式。但是FM模型只能发现两两特征组合的关系，并且需要穷举所有的特征组合，而其中包含部分没用的组合特征。后来发展出来了使用深度神经网络去自动挖掘更高层次的特征组合关系。但其实在使用神经网络之前，GBDT也是一种经常用来发现特征组合的有效思路。

$$y(\mathbf{x}) = \underbrace{w_0 + \sum_{i=1}^{n} w_i x_i}_{\text{线性}} + \underbrace{\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_{i,f_j}, \mathbf{v}_{j,f_i} \rangle x_i x_j}_{\text{非线性（交叉特征）}}$$

Facebook 2014年的文章介绍了通过GBDT解决LR的特征组合问题。具体的融合方案可以通过下面的例子解释。

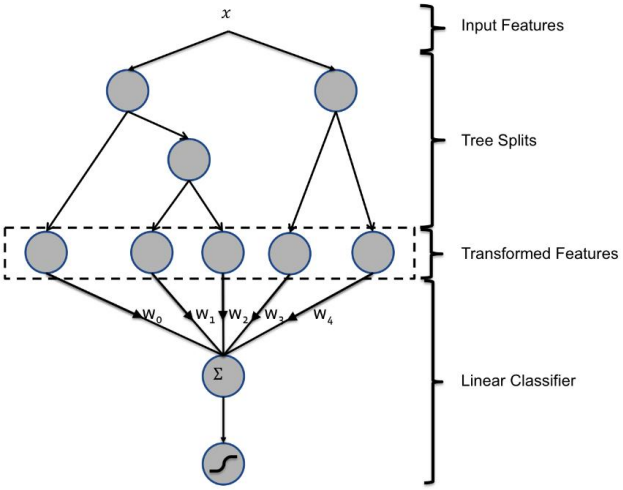


图 4-1：GBDT+LR 示意图

图中的两棵树为 GBDT 拟合的两棵树，GBDT 是每次都在之前构建的决策树的残差上构建下一棵决策树。x 为一个输入样本，x 样本分别落到每棵树的叶子节点上，每个叶子节点对应 LR 一维特征。左树有三个叶子节点，右树有两个叶子节点，最终的特征即为五维的向量。对于输入 x，假设他落在左树第一个节点，编码[1,0,0]，落在右树第二个节点则编码[0,1]，所以整体的编码为[1,0,0,0,1]，这类编码作为特征，输入到 LR 中进行分类。

4.2.2 模型设计的思路

模型与特征工程都对最终结果有较大影响，GBDT 的作用体现在特征工程上面。我们可以把 GBDT 部分看作一种有监督的特征编码，从根节点到叶子节点的一条路径，表示的是特征上的一个特定的规则，叶子节点的编号代表了这种规则，表征了样本中的信息，而且进行了非线性的组合变换。所以，GBDT 的作用就是对特征进行组合，自动帮助我们找到有意义的特征组合。

之所以使用集成的决策树模型，而不是单棵的决策树模型：一棵树的表达能力很弱，不足以表达多个有区分性的特征组合，多棵树的表达能力更强一些。可以更好的发现有效的特征和特征组合。

之所以采用 GBDT 而非随机森林：GBDT 前面的树，特征分裂主要体现对多数样本有区分度的特征；后面的树，主要体现的是经过前 N 颗树，残差仍然较大的少数样本。优先选用在整体上有区分度的特征，再选用针对少数样本有区分度的特征，思路更加合理。

4.2.3 模型实现与结果分析

GBDT 和逻辑回归均借助 python 中的 sklearn 模块。首先将训练集按照 7：3 切分成训练集和测试集，再把训练集切分成两个部分，一部分训练 GBDT 模型，另一部分输入到训练好的 GBDT 模型生成特征，然后作为特征输入到逻辑回归模型。把训练集分成两个部分，是为了防止过拟合，如果是 GBDT 和 LR 都用同样的数据训练，那么 LR 的输入特征中已经包含了标签的信息，这是不合理的。

最后在测试集上评价模型，画 ROC 曲线，横坐标为假正率，纵坐标为真正率，AUC 值为 0.796.

对于二分类问题，我们也可以用精确率、召回率和 F1 值来评价效果。对于评分低的类别的预测情况，精确率为 0.63，召回率为 0.60，F1 值为 0.62；评分高的类别的预测情况，精确率为 0.81，召回率为 0.82，F1 值为 0.82.可见效果不是很好。所以后面尝试了其他模型。

表 4-1：GBDT+LR 预测结果

	precision	recall	F1 score
False	0.63	0.60	0.62
True	0.81	0.82	0.82

4.3 GBDT+DeepFM

4.3.1 模型介绍

前面提到的 FM 模型考虑了二阶特征的组合，为了挖掘高阶特征，一个自然的想法是引入神经网络部分。DeepFM 就是集成了 FM 和 DNN 的神经网络框架，分别负责低阶特征的提取和高阶特征的提取。这两部分共享同样的输入。DeepFM 的预测结果可以写成：

$$\hat{y} = sigmoid(y_{FM} + y_{DNN})$$

$$y_{FM} = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} x_i x_j$$

深度部分是一个前馈神经网络，因为输入一般是稀疏的，所以加入一个嵌入层将输入向量压缩到低维稠密向量。压缩到低维向量的权重是 FM 和神经网络共享的。

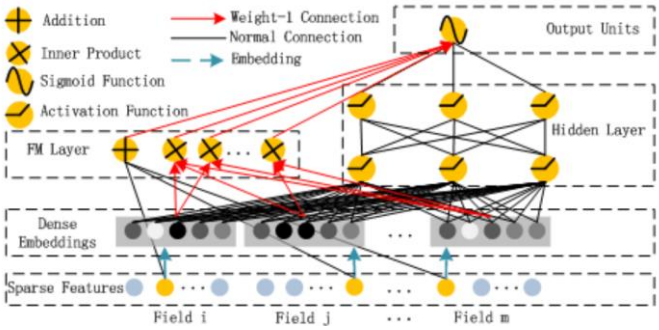


图 4-2：DeepFM 示意图

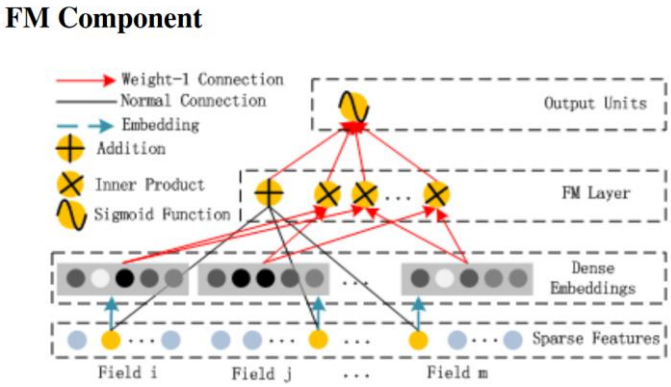


图 4-3：FM 部分

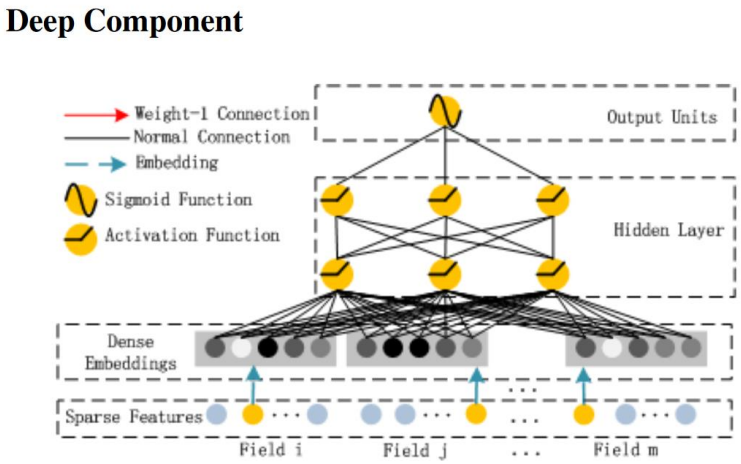


图 4-4：Deep 部分

4.3.2 模型实现与结果分析

模型用 tensorflow 实现。主要分为四个部分，首先是 FM 中的一阶部分，主要是将特征分别乘上对应的系数。

```
# ---------- first order term ----------
##初始化wij
feature_bias = tf.Variable(tf.random_uniform([feature_size, 1], 0.0, 1.0), name="feature_bias_0")
y_first_order = feature_bias
## wij * xij
y_first_order = tf.reduce_sum(tf.multiply(y_first_order, feat_value), 2)  # None * F
## 增加dropout，防止过拟合
y_first_order = tf.nn.dropout(y_first_order, dropout_keep_fm[0]) # None * F
```

第二部分是 FM 的二阶部分，主要是对x_i和x_j两两组合，并找到它们分别对应的特征向量。

```
# ---------- second order term ---------------
#vi*xi
embeddings = tf.multiply(embeddings, feat_value)
# 和平方
summed_features_emb = tf.reduce_sum(embeddings, 1)  # None * K
summed_features_emb_square = tf.square(summed_features_emb)  # None * K

# 平方和
squared_features_emb = tf.square(embeddings)
squared_sum_features_emb = tf.reduce_sum(squared_features_emb, 1)  # None * K

# 和平方与平方和按公式组合
y_second_order = 0.5 * tf.subtract(summed_features_emb_square, squared_sum_features_emb)  # None * K
y_second_order = tf.nn.dropout(y_second_order, dropout_keep_fm[1])  # None * K
```

第三部分是全连接神经网络。其中添加 Dropout 防止过拟合。

第四部分是把 DeepFM 和 FM 的输出拼接，然后 sigmoid 函数转换成最后的得分。

```
# ---------- DeepFM ----------
with tf.name_scope("deepfm"):
    concat_input = tf.concat([y_first_order, y_second_order, y_deep], axis=1)
    if MODETYPE==0:##deepfm
        concat_input = tf.concat([y_first_order, y_second_order, y_deep], axis=1)
        input_size = feature_size + embedding_size + deep_layers[-1]
    elif MODETYPE==1:##fm only
        concat_inpMODETYPEut = tf.concat([y_first_order, y_second_order], axis=1)
        input_size = feature_size + embedding_size
    elif MODETYPE==2:##dnn only
        concat_input = y_deep
        input_size =  deep_layers[-1]

    glorot = np.sqrt(2.0 / (input_size + 1))
    weights["concat_projection"] = tf.Variable(np.random.normal(loc=0, scale=glorot, size=(input_size, 1)),
                    dtype=np.float32 ,name="concat_projection0")  # layers[i-1]*layers[i]
    weights["concat_bias"] = tf.Variable(tf.constant(0.01), dtype=np.float32 ,name="concat_bias0")
    out = tf.add(tf.matmul(concat_input, weights["concat_projection"]), weights["concat_bias"],name='out')

score=tf.nn.sigmoid(out,name='score')
```

最后，用 tensorboard 查看模型在测试集上的 AUC 的情况。DeepFM 在 1950 次迭代后，AUC 为 0.865.与 FM 比较，FM 在 1950 次迭代后，AUC 为 0.8106。可以看出 DeepFM 的效果要好于 FM 模型。

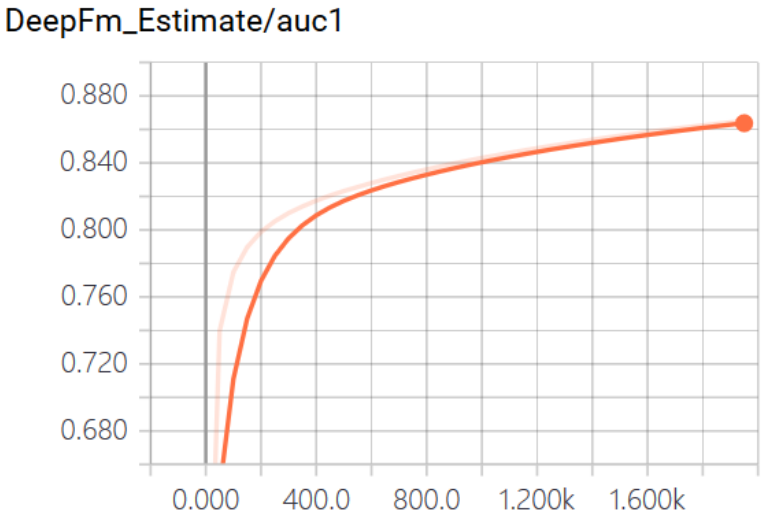


图 4-5：DeepFM 的 AUC 变化情况

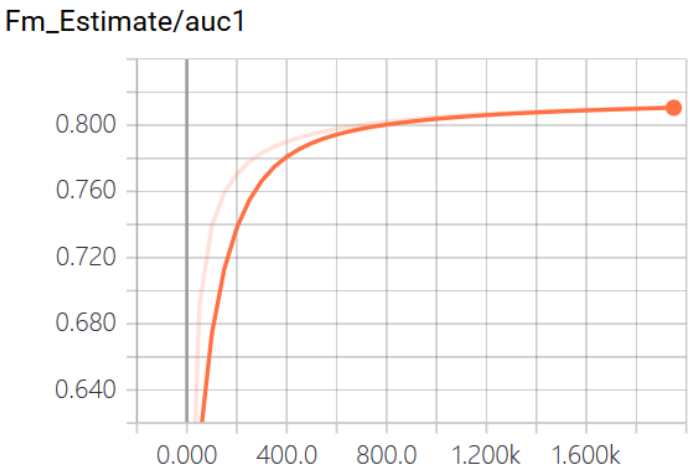


图 4-6：FM 的 AUC 变化情况

4.4 小结

尝试了利用机器学习和深度学习模型做推荐，具体模型包括 GBDT+LR，FM 以及 GBDT+DeepFM。最终结果为 GBDT+DeepFM 的效果较好。

5. 模型小结

5.1 浅层模型与深层模型的结合

我们把浅层与深层推荐模型综合起来，数据不同，处理流程不同，当没有用户和电影的内容信息时，我们用浅层模型；当没有评分数据，比如用户没有评分习惯时，我们用深度模型；当评分与内容数据都有时，我们先用深度模型初筛出用户喜欢的电影，然后用浅层模型做更精细的预测和排序。

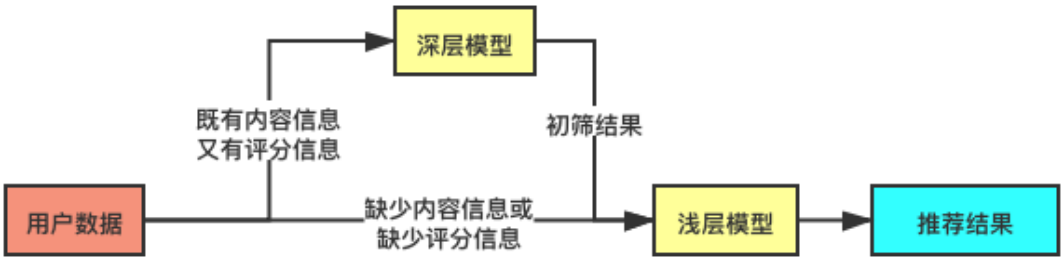


图 5-1：浅层模型与深层模型结合示意图

5.2 模型评价与展望

5.2.1 浅层模型

浅层模型综合了 4 个协同过滤模型的结果，模型效能更加全面稳健，二分类的 AUC 与余弦相似度分别有 0.704 与 0.855，相比于基础值 0.5 与 0.1004 有了较大的提升；同时浅层模型中相似度的计算也结合了内容特征，比一般只基于评分数据训练的模型更加精细化。但由于计算机性能受限，在模型只用了 5000 个用户对 3000 部电影的评分；同时，模型的调参过程也可以更加精细化。

5.2.2 深层模型

深层模型改进了原有的 DeepFM 和 GBDT+LR 模型，模型的优点一为 DeepFM 的输入特征个数可通过 GBDT 的基模型个数以及每棵树的节点个数来调整，适用于计算资源不足时需要减少 DeepFM 输入特征个数的场景。优点二为实验的 AUC 值比 GBDT+LR 和 FM 均有提升。

6. 参考文献

[1]He X , Bowers S , Candela J Q , et al. [ACM Press 20th ACM SIGKDD Conference - New York, NY, USA (2014.08.24-2014.08.27)] Proceedings of 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining - ADKDD\"14 - Practical Lessons from Predicting Clicks on Ads at Facebook[C]// Eighth International Workshop on Data Mining for Online Advertising. ACM, 2014:1-9.

[2]Guo H , Tang R , Ye Y , et al. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction[J]. 2017.

[3]黄昕，推荐系统与深度学习[M].北京：清华大学出版社，2019：163-171.

[4]GBDT+LR 融合方案实战. https://www.jianshu.com/p/96173f2c2fb4,2019.11.22

[5]FFM 算法解析及 python 实现. https://www.cnblogs.com/wkang/p/9788012.html，2019.11.22