**NeuMF调参实验**

1. **前置认知**

本次实验在电影评分数据集ml-1m的基础上，针对Top-N推荐任务，选用评价指标NDCG作为skopt自动调参工具的最大化学习目标，从而完成NeuMF模型的调参工作。

* 1. **数据集**

本次实验所涉及的数据，如下表所示。具体而言，train中每对（user\_id,item\_id）处，取值为1表示某用户对某物品评过分（即：认定为感兴趣），取值为0表示某用户对某物品没评过分。此后，在构建训练集train\_x和标签labels的过程中，每当遍历到一个取值为1的用户-物品对时（train有994169个评过分的用户-物品对），除了需要在训练集train\_x中存储该正例样本外，还需要追加存储针对该用户的num\_negatives个负例样本（该用户-负例物品对，不应该在train中取值为1）。

针对6040个用户的Top-N推荐测试任务，testRatings用于提供一个对应每个测试用户的正例标签物品item\_id，testNegatives用于提供对应每个测试用户的若干负例物品。至此，针对每个测试用户，可根据对应的若干负例物品和一个正例标签物品，生成有序的推荐列表。值得注意的是，为了保证推荐的有效性，某测试用户在testNegatives中的若干负例物品和在testRatings中的一个正例物品，都不应该在train中对应的用户-物品对处取值为1。

**表1 调参实验使用到的数据及其解释**

|  |  |  |  |
| --- | --- | --- | --- |
| **数据名称** | **维数** | **行示例** | **注释** |
| train | 6040\*3706 | len([0, 1, 0, ..., 1])=3706 | 用户-物品交互矩阵 |
| testRatings | 6040\*2 | [user\_id, item\_id] | 测试用户的正例标签物品 |
| testNegatives | 6040\*99 | [item\_id1, ..., item\_id99] | 测试用户的负例候选物品 |
| train\_x | [994169\*(num\_negatives+1)]\*2 | [user\_id, item\_id] | 训练集 |
| labels | 994169\*(num\_negatives+1) | [0, 1, 0, ..., 1] | 训练集标签 |

* 1. **自动调参工具skopt**

在自定义的搜索空间中，Scikit-optimize利用贝叶斯优化尝试不同的参数组合，并通过比较它们在自定义目标函数中的最终返回值，从而获得相对最优参数组合。相较于只能通过简单枚举参数值所有可能组合的sklearn.model\_selection.GridSearchCV，skopt能够利用之前测试过的参数组合及其结果，在搜索空间中针对性地选取新参数组合用于测试，从而极大地提高了调参效率。同时，自定义的搜索空间，也能够给开发者提供需调整哪些参数及其取值范围的灵活性。此外，skopt封装有关于评估结果的可视化方法，能够直观地体现不同参数在其取值范围内的性能表现。

关于skopt的使用方法，如下所示：

1. **定义搜索空间**

在搜索空间中，主要定义需要调哪些参数以及参数值的调整范围。具体代码，如下所示。其中，Integer()表示整数，Real()表示浮点数。

space = [  
 Integer(1, 80, name='topK'),  
 Integer(4, 100, name='num\_factors'),  
 Integer(4, 100, name='num\_negatives'),  
 Real(1e-4, 1e-2, prior='log-uniform', name='lr'),  
 Integer(5, 25, name='epochs'),  
 Integer(8, 256, name='layers\_1'),  
 Integer(8, 256, name='layers\_2'),  
 Real(0, 0.5, name='dropout'),  
]

1. **定义优化器参数**

具体代码，如下所示。其中，n\_calls表示本次程序运行需要尝试多少个参数组合，n\_random\_starts表示优化器在寻找最佳位置前的随机探索迭代数，base\_estimator用于指定拟合目标函数的基本估计器，acq\_func用于指定采集函数，random\_state用于指定随机种子。

HPO\_PARAMS = {  
 'n\_calls': 18,  
 'n\_random\_starts': 1,  
 'base\_estimator': 'ET',  
 'acq\_func': 'gp\_hedge', # 采集函数，概率选择EI、PI和LCB中的一个  
 'random\_state': 15 # 固定随机数种子，使结果可重现  
}

1. **定义目标函数**

具体代码，如下所示。在目标函数的内部，需要完成：定义数据迭代器、模型实例化和模型训练与测试评估等工作，并最终返回属于标量的最小化目标值（比如loss或-NDCG等）

@use\_named\_args(space)  
def objective(topK, num\_factors, num\_negatives, lr, ...):# 1. DataLoader  
 # 2. 显示当前评估的参数组合  
 print(f"Evaluating: topK={topK}, num\_factors={num\_factors}, num\_negatives={num\_negatives}, lr={lr}, ...")  
 # 3. 模型实例化  
 # 4. 模型训练与测试，获得min\_target\_value  
 return min\_target\_value # 5. 返回最小化目标值

1. **调参测试**

具体代码，如下所示。本次选用基于高斯过程的贝叶斯优化算法gp\_minimize，还有dummy\_minimize、forest\_minimize和gbrt\_minimize等优化算法可选，具体解释详见：

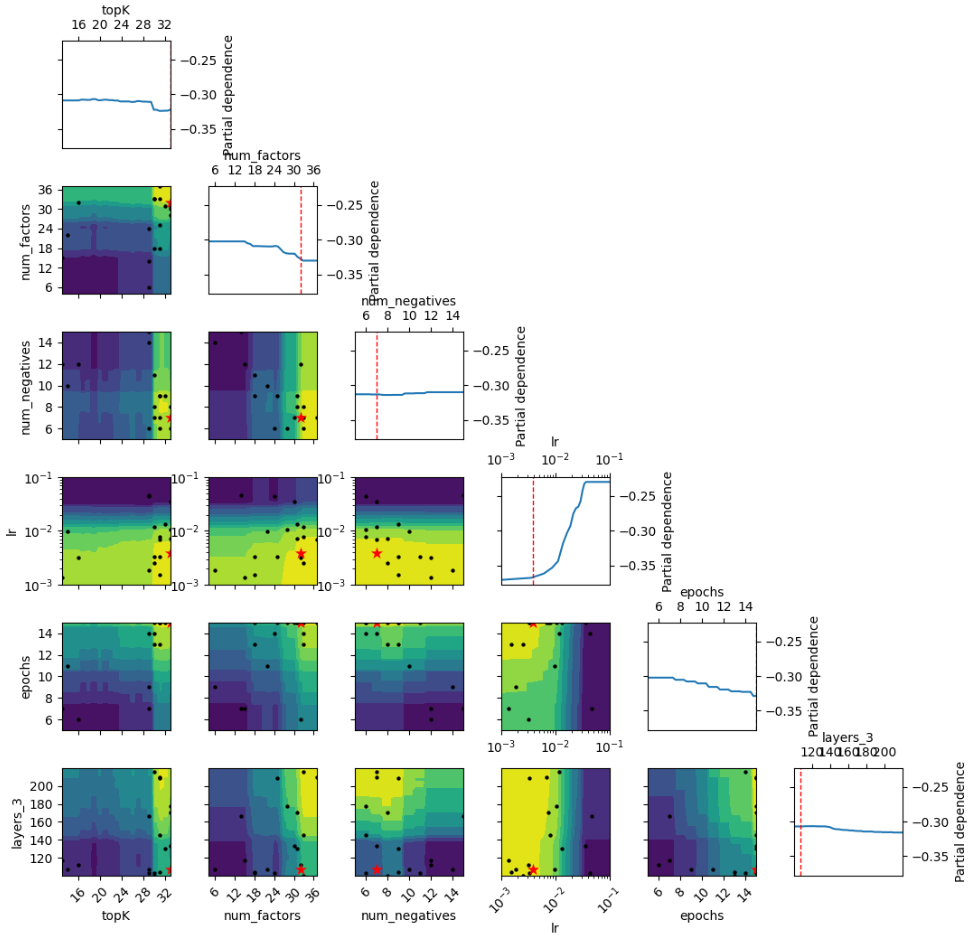
[一个超参数优化工具：Scikit Optimize-CSDN博客](https://blog.csdn.net/lemonbit/article/details/125013118?spm=1001.2014.3001.5506)

old\_results = load\_results(file\_path) # 加载历史调参结果  
if old\_results is not None: # 在上次调参结果的基础上，继续进行调参测试  
 print("继续测试")  
 print("old\_results.x\_iters: ", old\_results.x\_iters)  
 print("old\_results.func\_vals: ", old\_results.func\_vals)  
 results = gp\_minimize(objective, space,  
 x0=old\_results.x\_iters,  
 y0=old\_results.func\_vals,  
 \*\*HPO\_PARAMS)  
else:  
 print("从零开始测试") # 从新开始调参测试  
 results = gp\_minimize(objective, space, \*\*HPO\_PARAMS)

1. **结果可视化**

具体代码，如下所示。可完成调参结果的可视化，并在控制台中输出最佳参数组合。

save\_results(results, file\_path) # 保存结果  
print("results.x\_iters：", results.x\_iters)  
print("results.func\_vals：", results.func\_vals)  
skopt.plots.plot\_objective(results) # 结果可视化  
plt.show()  
best\_params = to\_named\_params(results, space) # 打印最佳参数组合  
print("Best parameters:", best\_params)



1. **Base\_NeuMF调参实验**

令人遗憾的是，直接运行来自网络的NeuMF代码，其在测试集上的最佳平均NDCG值仅为**0.32**。由此，可以直接指定-NDCG为skopt在调参过程中的最小化优化目标，来获得最大化NDCG评估值的相对最优参数组合，从而完成Base\_NeuMF模型的调参工作。

* 1. **Base\_NeuMF网络结构**

**表2 Base\_NeuMF网络结构及解释**

|  |  |  |
| --- | --- | --- |
| **网络部分** | **具体层** | **解释** |
| 嵌入部 | MF\_Embedding\_User  MF\_Embedding\_Item  MLP\_Embedding\_User  MLP\_Embedding\_Item | GML和MLP分别完成用户和物品的嵌入表示 |
| GML | mf\_vec = torch.mul | 逐元素相乘，合并用户和物品的MF嵌入表示 |
| MLP | torch.cat  (linear+F.relu)\*3  mlp\_vec = self.linear(x) | 将用户和物品的MLP嵌入表示拼接在一起， 投喂到多层感知机中。 |
| 输出部 | torch.cat(mf\_vec, mlp\_vec)  (Linear+sigmoid)\*1 | 将GML和MLP部分的输出向量拼接起来， 投喂到输出层中。 |

* 1. **搜索空间**

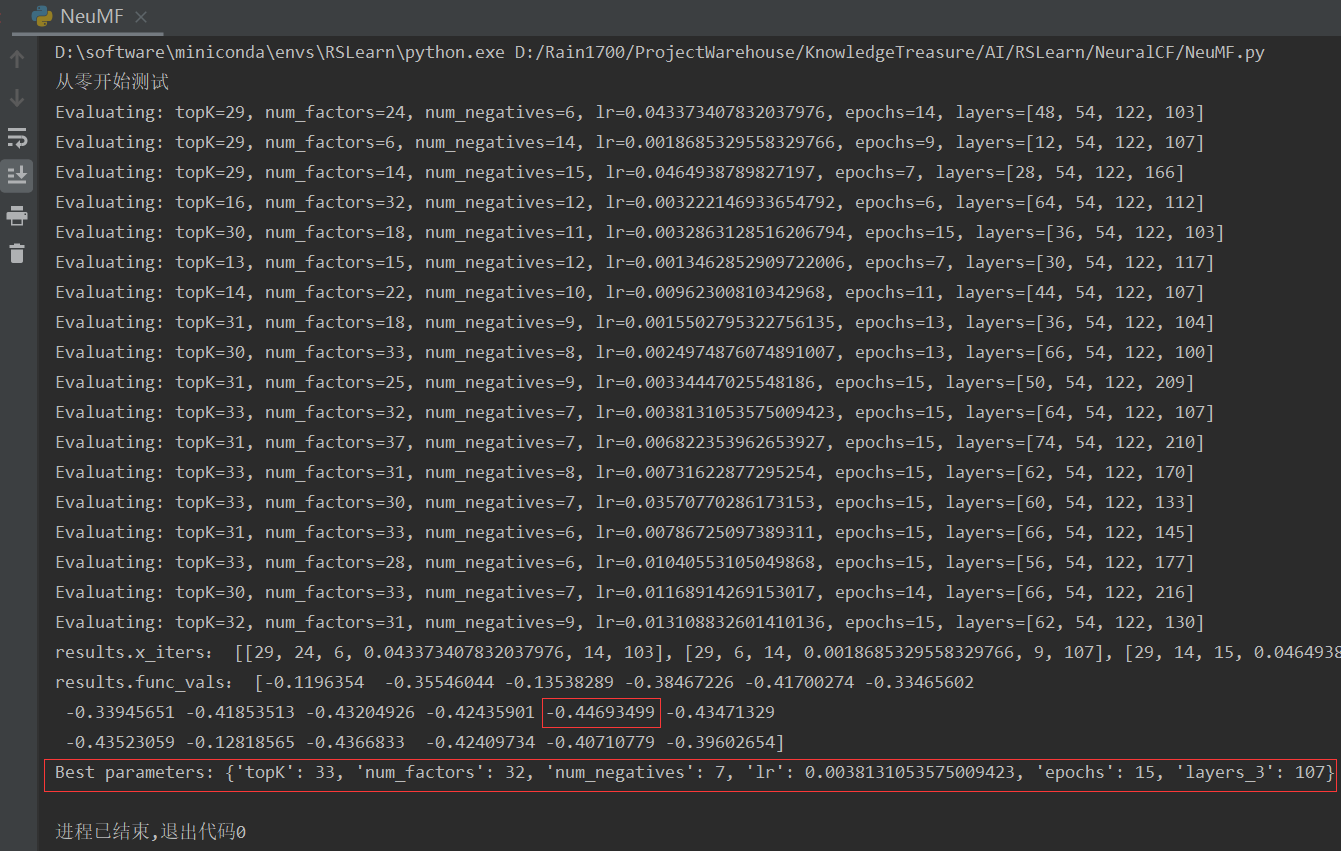
考虑到调参时间，需要在预实验的基础上，进一步确定需要调哪些参数及其搜索范围。

**表3 正式调参实验的搜索空间**

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **取值范围** | **是否调参** | **解释** |
| topK | (13, 33) | √ | 推荐物品的数量；预实验最优值为13 |
| num\_factors | (4, 37) | √ | GML嵌入维数；预实验最优值为37 |
| num\_negatives | (5, 15) | √ | MLP嵌入维数及输出层；预实验最优值为5 |
| lr | (1e-3, 1e-1) | √ | 学习率；预实验最优值为0.0057 |
| epochs | (5, 15) | √ | 迭代周期；预实验最优值为5 |
| layers\_1 | 固定为54 |  | MLP隐藏层1；预实验中影响很小 |
| layers\_2 | 固定为122 |  | MLP隐藏层2；预实验中影响很小 |
| layers\_3 | (100, 220) | √ | MLP隐藏层3；预实验最优值为114 |

* 1. **调参实验结果**

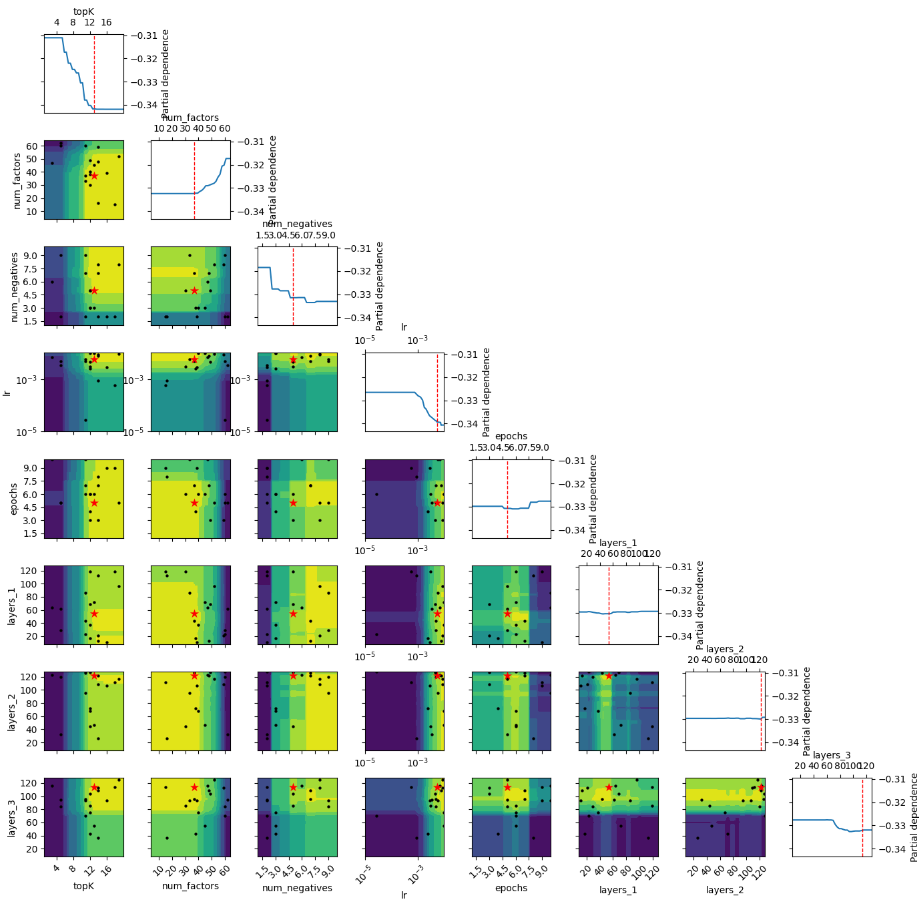
本次调参实验历经13小时左右，在前述自定义搜索空间基础上，总共测试了18个参数组合。完成测试的每个参数组合及最佳平均NDCG，具体如下图所示。其中，最佳参数为：topK=33、num\_factors=32、num\_negatives=7、lr=0.0038、epochs=15、layers\_1=54、layers\_2=122、layers\_3=107，在测试集上的最佳平均NDCG值约为**0.447**。



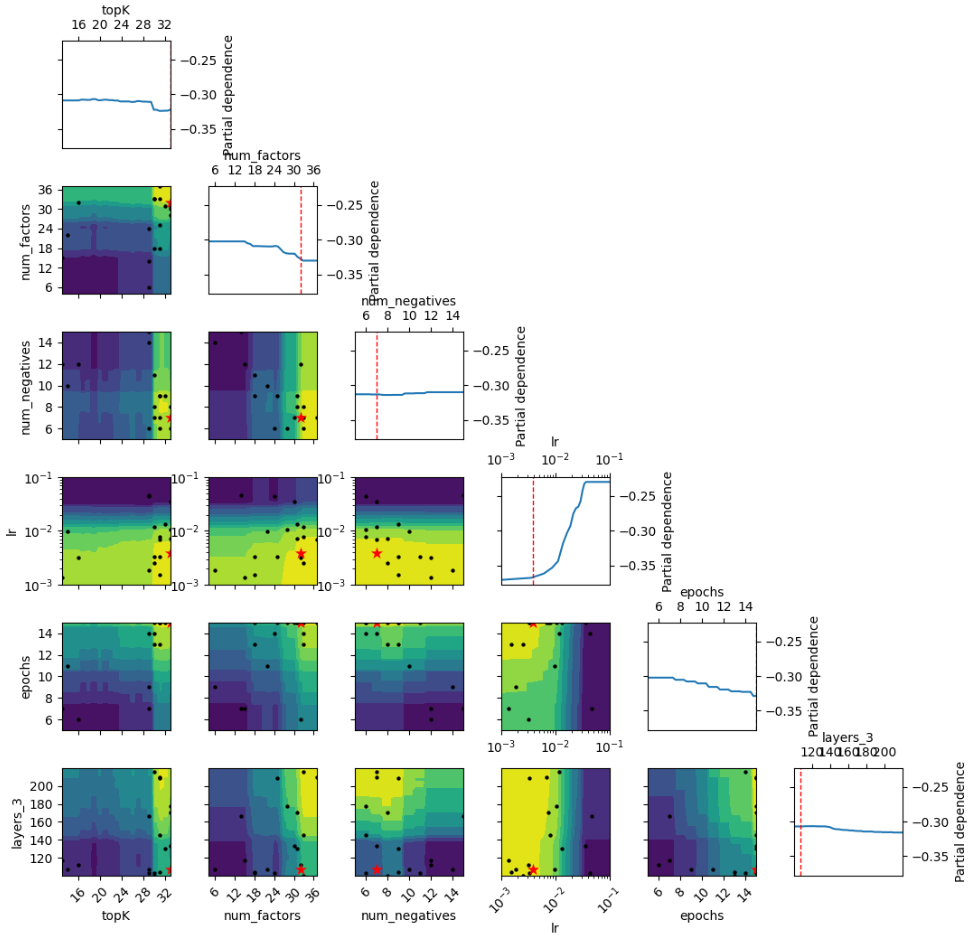
**图1. Base\_NeuMF的调参实验结果**

* 1. **结果分析**

关于预实验和正式调参实验的可视化结果，分别如图2、3所示。



**图2. Base\_NeuMF的预调参实验结果可视化**



**图3. Base\_NeuMF的正式调参实验结果可视化**

从各参数在其取值范围内的性能变化趋势来看，可总结出如下结论：

1. topK的更大取值（>33）或许能提升性能。考虑到运行时间，可固定为33。
2. num\_factors的更大取值（>32）或许能提升性能。考虑到运行时间，可固定为32。
3. num\_negatives在（5, 15）区间内提升不明显。故此，可固定为5。
4. lr在0.0038左右的取值，能够最大限度地提升性能。故此，可固定为0.0038。

（5）epochs在（5, 15）区间内提升效果较小，更多的迭代周期并没有显著提升性能。

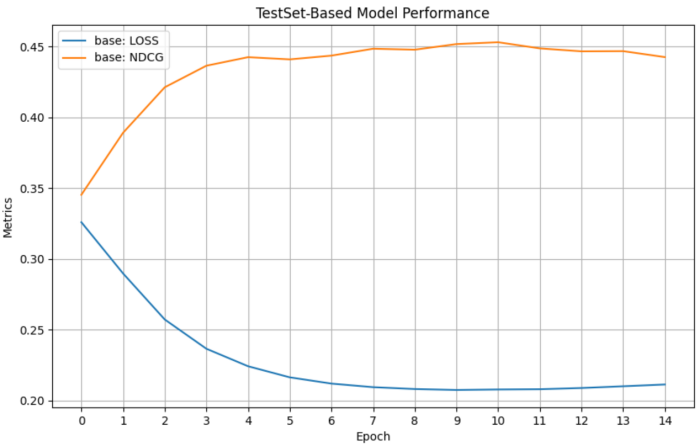
（6）layers\_3在（100, 220）区间内提升不明显。故此，可固定为100。

（7）layers\_2在（8, 128）区间内提升不明显。故此，考虑预实验最优值，可固定为122。

（8）layers\_1在（8, 128）区间内提升不明显。故此，考虑预实验最优值，可固定为54。

**2.5 最终实验**

参考前述结论的同时兼顾运行时间，可使用如下参数组合进行最终实验：topK=33、num\_factors=32、num\_negatives=5、lr=0.0038、epochs=15、layers\_1=54、layers\_2=122、layers\_3=100、batch\_size=128，可得出其在测试集上的最佳平均NDCG值约为**0.45**。训练损失和在测试集上的NDCG评估值变化，具体如下图所示。



**图4. 最终实验结果可视化**

1. **GRU\_NeuMF调参实验**

针对前述Base\_NeuMF做了大量调参工作后，虽然其在测试集上的最佳平均Hit Ratio值可达0.9左右，但是最佳平均NDCG值却仅有0.45左右。至此，若需要更多地考虑正例标签物品在推荐列表中的排名，Base\_NeuMF的学习能力显然不足。

因此，针对前述Base\_NeuMF模型，进行如下改动尝试：

1. GMF和MLP部分的嵌入表示维数不再相同；
2. MLP部分的用户和物品嵌入表示维数不再相同；
3. 自定义nn.GRUCell实现类，并在NeuMF实现类中：将GRU模块同MLP部分串联；
4. 将MLP部分的线性层增加到6层，并在每个线性层后额外追加nn.Dropout。
   1. **搜索空间**

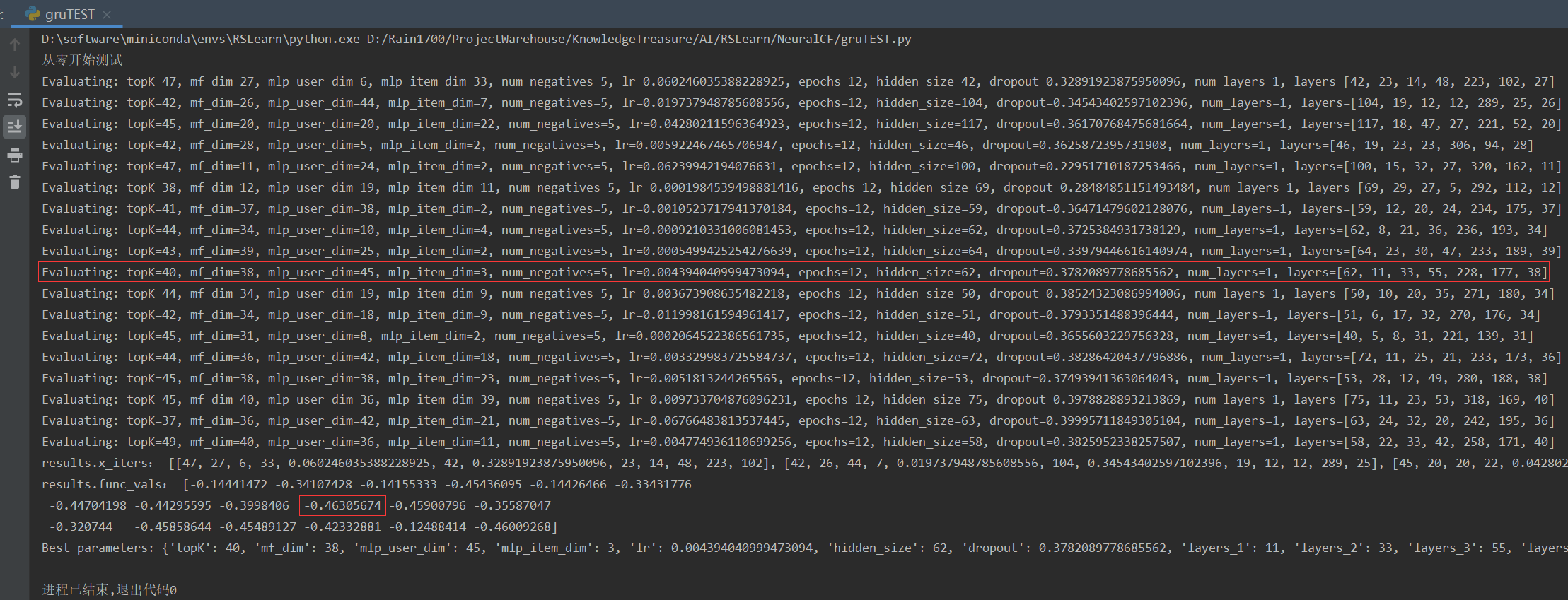
考虑到调参时间，需要事先根据切片而来的小数据集进行预实验，从而筛选出需要调试哪些参数以及对应的搜索范围。

**表4 正式调参实验的搜索空间**

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **取值范围** | **是否调参** | **解释** |
| topK | (33, 50) | √ | 推荐物品的数量； |
| mf\_dim | (8, 40) | √ | GML嵌入维数； |
| mlp\_user\_dim | (1, 45) | √ | MLP用户嵌入维数； |
| mlp\_item\_dim | (1, 40) | √ | MLP物品嵌入维数； |
| lr | (1e-4, 1e-1) | √ | 学习率； |
| hidden\_size | (40, 128) | √ | GRU隐藏状态维数； |
| dropout | (0, 0.4) | √ | 正则化手段；预防过拟合，故保留 |
| layers\_1 | (4, 30) | √ | MLP隐藏层1；预实验倾向于小区间 |
| layers\_2 | (4, 50) | √ | MLP隐藏层2；预实验倾向于小区间 |
| layers\_3 | (4, 55) | √ | MLP隐藏层3；预实验倾向于小区间 |
| layers\_4 | (220, 320) | √ | MLP隐藏层4；预实验倾向于大区间 |
| layers\_5 | (4, 200) | √ | MLP隐藏层5；趋势不明朗，但保留 |
| num\_negatives | 5 |  | 训练集负例追加数；局部最优值，故固定 |
| epochs | 12 |  | 迭代周期；大迭代周期效果不理想，故固定 |
| num\_layers | 1 |  | GRU层堆叠数；更多堆叠效果不佳，故固定 |

* 1. **调参实验结果**

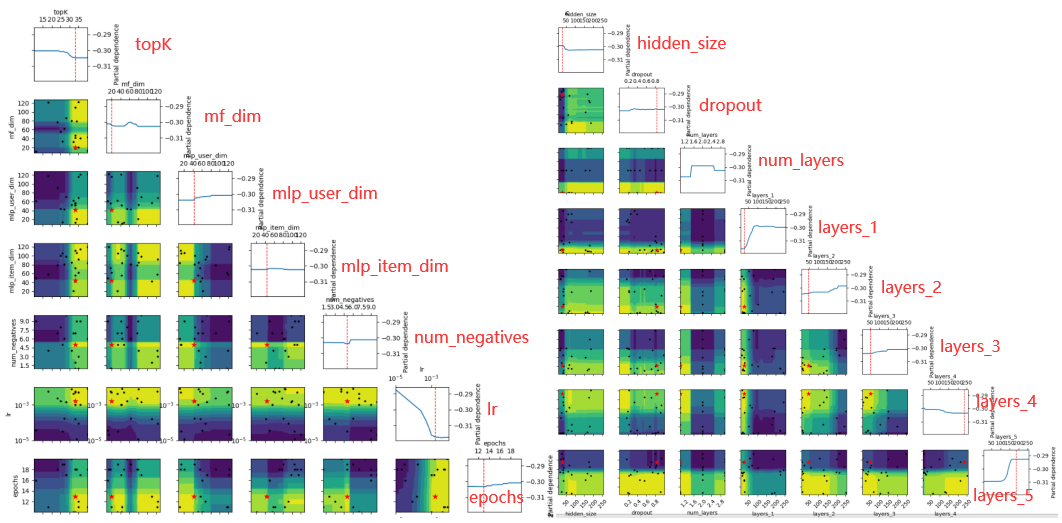
本次调参实验，根据前述自定义的搜索空间，总共测试了18个参数组合。完成测试的每个参数组合及最佳平均NDCG，具体如下图所示。其中，最佳参数为：topK=40、mf\_dim=38、mlp\_user\_dim=45、mlp\_item\_dim=3、lr=0.00439、hidden\_size=62、dropout=0.3782、layers\_1=11、layers\_2=33、layers\_3=55、layers\_4=228、layers\_5=177、batch\_size=128、num\_negatives=5、num\_layers=1、epochs=12，在测试集上的最佳平均NDCG值约为**0.463**。



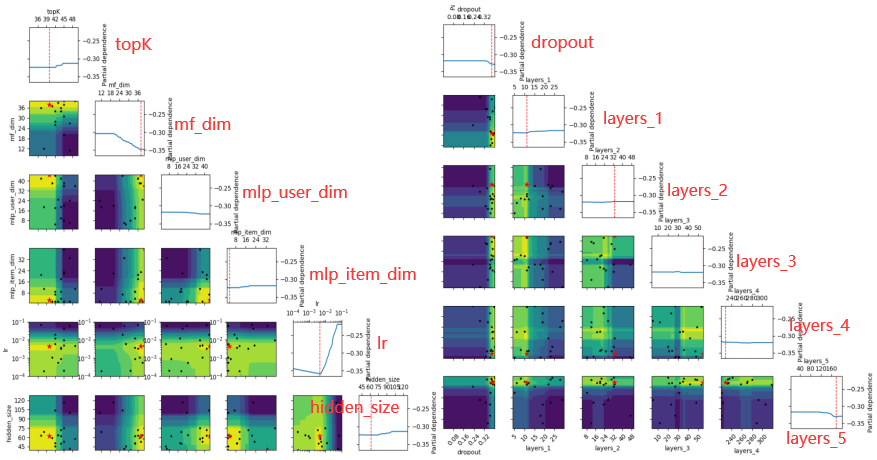
**图5. GRU\_NeuMF的调参实验结果**

* 1. **结果分析**

关于预实验和正式调参实验的可视化结果，分别如图6、7所示。



**图6. GRU\_NeuMF的预调参实验结果可视化**



**图7. GRU\_NeuMF的正式调参实验结果可视化**

从各参数在其取值范围内的性能变化趋势来看，可总结出如下结论：

（1）topK在（33, 40）区间内达到最佳且变化趋势稳定。故此，可固定为40。

（2）mf\_dim的更大取值（>38）或许能提升性能。考虑到运行时间，可固定为38。

（3）mlp\_user\_dim提升趋势不明显。故此，可固定为45。

（4）mlp\_item\_dim提升趋势不明显。故此，可固定为3。

（5）lr在0.00439处，性能提升效果最为显著。故此，可固定为0.00439。

（6）hidden\_size在（40，70）区间内相对最佳。参考实验获得的最佳参数，可固定为62。

（7）dropou的更大取值（>0.378）或许能提升性能。考虑到调参时间，可固定为0.378。

（8）layers\_1提升趋势不明显。故此，可固定为11。

（9）layers\_2提升趋势不明显。故此，可固定为33。

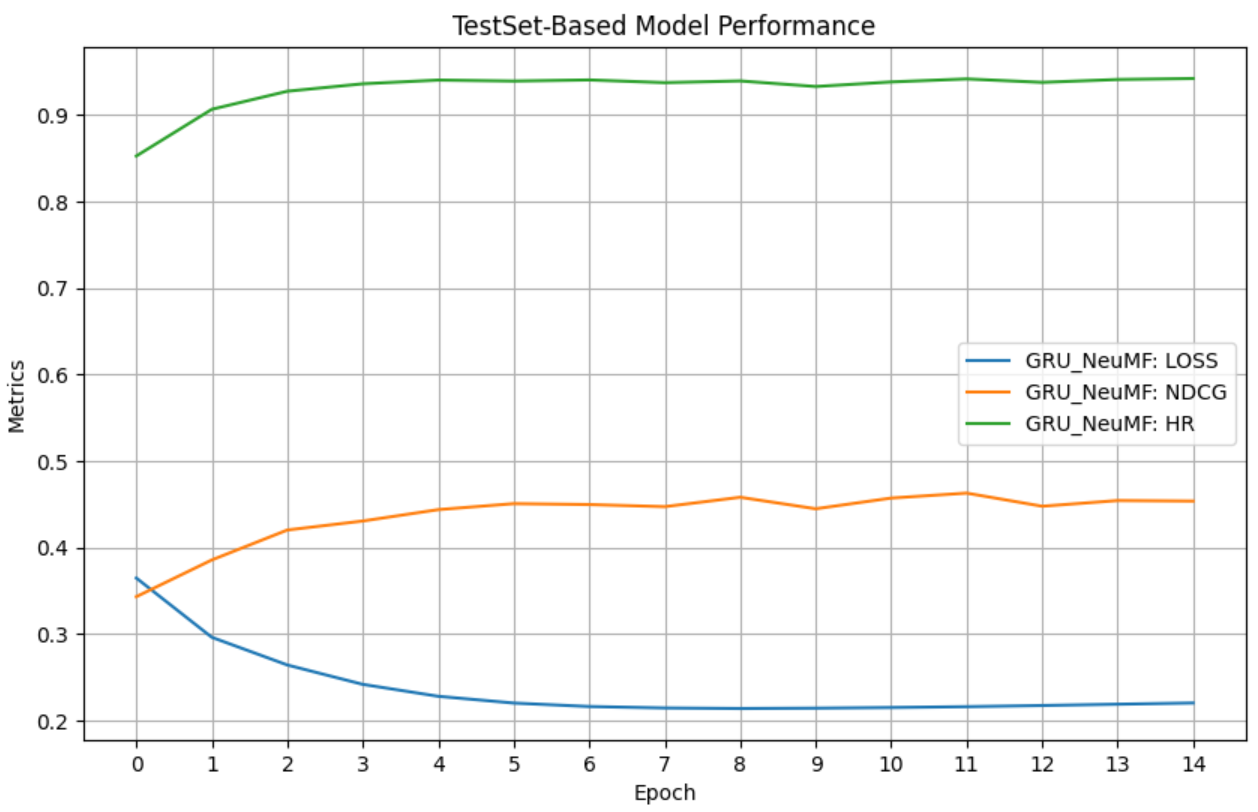
（10）layers\_3提升趋势不明显。故此，可固定为55。

（11）layers\_4提升趋势不明显。故此，可固定为228。

（12）layers\_5提升趋势不明显。故此，可固定为177。

* 1. **最终实验**

参考前述结论中固定的参数值，batch\_size可分别取64、128、256和512，进行最终的对比测试实验。由此，获得的最佳参数组合为：topK=40、mf\_dim=38、mlp\_user\_dim=45、mlp\_item\_dim=3、lr=0.00439、hidden\_size=62、dropout=0.3782、layers\_1=11、layers\_2=33、layers\_3=55、layers\_4=228、layers\_5=177、epochs=15、num\_layers=1、num\_negatives=5、batch\_size=128，可得出其在测试集上：最佳平均NDCG值约为**0.463**；最佳平均Hit Ration值约为**0.942**。训练损失和在测试集上的HR、NDCG评估值变化，具体如下图所示。



**图8. 最终实验结果可视化**

1. **总结**

一般来说，数据预处理相关参数（比如：批归一化等）、网络参数（比如：网络层数、正则化和激活函数等）及超参（比如：优化算法、学习率、迭代周期、批次数和衰减函数等），都属于调参工作的测试对象。当遭遇过拟合问题时：如若训练样本太少则可进行数据增强；可以使用早停法，在过拟合前及时停止训练；引入dropout等正则化手段；适当降低学习率；减少迭代周期数。当遭遇欠拟合问题时：可适当增加网络层数；使用非线性激活函数ReLU；增加迭代周期数。当不收敛时，则可能是数据预处理过程存在问题；或者网络设计存在问题，与数据集不匹配。

在本次调参实验中，考虑到调参效率，一般可以选择切片出小数据集用于测试。同时，为了更公平地比较不同参数组合的效果，尤其当数据预处理环节存在随机采样操作时，需要事先设定好固定的随机数种子。此外，在网络模型实现类的前向传播函数中的某位置处插入其它模块时，需要事先明确待插入位置处前后的数据流维数，以及插入模块输入输出的维数要求，这样才能利用view()、reshape()或permute()等数据维度转换工具，更高效地完成模块插入工作。当然，除了本次实验中选用串联的方式来缝合模块，还可以采用并联、嵌套包含或者串并联混合等方式，完成网络模型中不同模块的灵活组合。

更为直观的感受是，调参工作更强调个人经验，没有普适的调参策略用于应对特定情景，更多时候需要视情况具体分析。比如，提高迭代周期数，有时候会提高模型性能，有时候会造成过拟合问题，反而降低了模型性能。从本次实验经历的角度出发，调参测试工具skopt不失为调参工作的好帮手。然而，本次实验的不足之处在于：

1. **缺乏应对程序运行突然中断的手段，未实时保存测试过的参数组合及其结果；**
2. **没有考虑数据样式，盲目地插入GRU模块，并没有充分发挥GRU的序列学习能力；**
3. **没有进一步探究GRU\_NeuMF模型中四项改动尝试，分别对于模型性能的影响；**
4. **没有结合来自其它技术路线的推荐模型，提高NeuMF模型的性能；**
5. **仅使用一种数据集和有限的评价指标，用于评估模型的性能，缺乏信服力。**