答疑 | 基础架构篇+特征工程篇常见问题解答

你好，我是王喆。

到今天为止，基础架构篇和特征工程篇我们都学完了。这段时间，我收到了同学们的很多留言，也看到了大家在学习和实践过程中的很多疑问。今天这节课，我挑了 10 道典型的问题，想和你好好讨论一下，希望可以帮助你解决困惑。

实战项目安装、操作类的问题

我们在第 2 节课讲了 SparrowRecys 项目的安装方法，不过，我在留言区看到，大家在安装的时候，还是会遇到很多问题。这里我整理出了两类典型的问题，我们一起看看。

问题 1：因为没有项目经验，想知道把 SparrowRecys 项目 git clone 到本地之后，怎么运行这个 Maven project？

这里我再重新描述一下整个安装和执行的过程，详细来说一共有 6 步：

1. 安装 IDEA。到这个地址下载 IDE，安装 IDEA 后，打开 IDEA；

2. 在 IDEA 中打开项目。选择 File->Open-> 选择 git clone 到的项目根目录，就可以把项目导入到 IDEA；

3. 配置 maven project。我们在 IDEA 的项目结构树的 pom.xml 上点击右键，设置为 maven project（最新的 IDE 版本也可能不用）就可以了；

4. 配置 SDK。SparrowRecsys 使用了 Java8，Scala2.11 的编译环境，你可以在 File->Project Structure->Project 中配置 Java SDK，并在 Global Libraries 中配置 Scala SDK；

5. 运行推荐服务器。我们找到类文件 class RecSysServer（com.wzhe.sparrowrecsys.online.RecSysServer），右键点击 -> run；

6. 打开 SparrowRecsys 首页，在浏览器中输入http://localhost:6010/，当看到 SparrowRecSys 首页的时候，就说明你整个配置和安装成功了。

问题 2：在项目中没有找到“为你推荐页”，也没有看到一些项目介绍中提到的推荐算法，是我安装过程中出错了吗？

这里我再强调说明一下，你没有安装错，SparrowRecsys 这个项目是随着课程的进展逐渐完善起来的。所以如果在你学习的时候课程还未完结，SparrowRecsys 中可能会缺少课程还未进行到的模块。比如为你推荐这个功能是在课程的“推荐模型篇”中加入的，所以具体的内容我也会在之后的课程中再加入。但课程已经讲解过或提到过的部分，一定会在 SparrowRecsys 中有对应的实践代码。

课程相关的知识误区

除了安装问题之外，我还发现同学们在学习的过程中对某些知识点有疑惑，那下面我就帮同学们来解答一些典型的疑惑。

问题 3：网上资料的大部分观点认为协同过滤这样的传统方法应该是在召回层，但我们课程中把协同过滤放在了排序（精排）层，这是为什么呢？

这是个好问题。我们知道，五六年前的传统推荐系统不少还在使用协同过滤作为主排序模型。但这几年它就被慢慢淘汰了，排序层变成了以深度学习推荐模型为主的复杂模型。但因为协同过滤类算法比较简单，线上计算过程也很高效，比如矩阵分解之后可以进行 embedding 快速召回，所以放在召回层也完全适用。

在这门课程中，我们总结的推荐系统架构是一个比较经典的架构，但你也没必要认为它就是无法改变的真理。在实际应用场景之中，我希望你能根据业务特点灵活运用。

问题 4：像多模态或者是通过其它预训练方法得到的向量，直接加到推荐排序模型作为特征的话，感觉没什么效果，我理解是预训练学习的目标和排序学习目标并不一致。这个问题老师是怎么看的？

首先，我觉得这是一个很好的业务实践的问题。多模态指的是在推荐系统中引入视频、图片、语音等多种不同形式的数据和特征，希望来提升推荐效果。

在实际的业务应用里，确实存在多模态特征效果不强的问题。结合我的实践经验，我会觉得问题根源还是因为目前多模态的技术本质上还处于比较初期的阶段。

比如用一些 CV 的技术去处理视频图像，识别出一些物品，比如视频里有汽车、有树木、有人物之类。但你要说这些物品对于最终的推荐效果到底有没有影响，比如说视频中出现汽车到底对用户的点击率影响有多大，我觉得还是比较微弱。在视频推荐中，这可能远不及知名演员一个要素的影响大。

当然，我一直强调所有的效果都要跟业务场景紧密结合起来，所以多模态到底有没有作用，根本无法一概而论，还是跟你的使用方法和对

业务的理解强关联。比如在短视频推荐中，如果你能精确识别出视频中的明星是哪位，再用它作为推荐特征，我想肯定对最终的推荐效果有正向影响。

问题 5：对训练数据中的某项特征进行平方或者开方，是为了改变训练数据的分布。训练数据的分布被改变后，训练出来的模型岂不是不能正确拟合训练数据了？

这个也是一个常见的误区，如果你有这样的问题，说明你还没有弄明白特征的分布和训练数据的分布之间的关系。

对训练数据中的某个特征进行开方或者平方操作，本质上是改变了特征的分布，并不是训练数据的分布。特征的分布和训练数据的分布没有本质的联系，只要你不改变训练数据 label 的分布，最终预测出的结果都应该是符合数据本身分布的。因为你要预测的是 label，并不是特征本身。而且在最终的预测过程中，这些开方、平方的特征处理操作是在模型推断过程中复现的，本质上可以看作是模型的一部分，所以不存在改变数据分布的问题。

问题 6：“为了使 Graph Embedding 的结果能够表达网络的‘结构性’，在随机游走的过程中，我们需要让游走的过程更倾向于 BFS（Breadth First Search，宽度优先搜索）”。这里应该是 DFS 吧？并且同质性是使用 BFS。

这是第 7 讲中的一个知识点，这个疑问非常地常见，因为 BFS，DFS 与结构性、同质性的关系本身确实有一点反直觉。这也是我们在学习 Node2vec 模型的时候经常会有的问题，也推荐其他有疑问的同学关注一下。

在这里，我需要再强调一下，课程中的描述是完全正确的，也就是为了使 Graph Embedding 的结果能够表达网络的“结构性”，在随机游走的过程中，我们需要让游走的过程更倾向于 BFS；为了表达“同质性”，需要倾向于 DFS。我们一定要厘清它们之间的正确关系。

这里，我也直接把Node2vec 原论文中的论述贴在了文稿里，你直接参考原文，我觉得会理解得更深刻一些。

> We observe that BFS and DFS strategies play a key role in producing representations that reflect either of the above equivalences.
>
> In particular, the neighborhoods sampled by BFS lead to embeddings that correspond closely to structural equivalence.
>
> The opposite is true for DFS which can explore larger parts of the network as it can move further away from the source node u (with sample size k being fixed).
>
> In DFS, the sampled nodes more accurately reflect a macro-view of the neighborhood which is essential in inferring communities based on homophily.
>
> 参考译文：
> 我们观察到，BFS 和 DFS 策略在产生向量表达时发挥着关键的作用。特别是通过 BFS 采样得到的邻域节点使生成的相应 Embedding 更接近结构性一致。而对于 DFS 来说，情况恰恰相反，由于 DFS 可以进一步采样到远离节点 u（样本大小 k 固定）

> 的部分，因此可以探索更大范围的网络。在 DFS 中，采样的节点可以更准确地反映邻域的宏观视图，这对于推断社区的同质性至关重要。

关于推荐系统的深入思考

解决了一些常见的知识性的疑问，我们再来看看一些关于课程具体内容的延伸思考。我觉得这些问题都提得都很好，说明同学们学习的时候都有在认真思考，同时，我也鼓励大家都带着问题来学习，把自己的思考分享出来，这也能帮助到更多的同学。

问题 7：老师，我注意到 Flink 最近更新比较频繁，号称可以做到流批一体分析，甚至 ETL 领域好像也可以用起来。那我们在设计系统架构的时候直接用 Flink 取代 Spark，让 ETL 和实时部分统一到一个架构上是否可行呢？

其实这也是大数据工程师们一直追求的批流一体的 Kappa 架构。

在 Kappa 架构的实践中，工程师们遇到的困难也不少。一是一些历史遗留问题，比如当前很多公司的数据体系大部分是建立在 Spark 基础上的，直接用 Flink 完全替代肯定有风险，所以很多公司还沿用着批流混合的 Lambda 架构。

另外是 Spark 和 Flink 发展的问题，Flink 在进化的同时 Spark 也在发展，比如 Spark 最近发展的 Structured Streaming 就是为了跟 Flink 竞争，而且 Spark 本身的社区成熟程度和这么多年的积累还是超过目前的 Flink 的，所以也难说 Flink 会完全替代 Spark。

但毫无疑问，批流一体是未来的方向，业内的工程师们也都在往这个方向努力。但我个人觉得 Spark 和 Flink 会长期共存、共同发展。

问题 8：老师，请问关于大数据数据出口的那部分内容，请问实时的用户推荐请求也是会先经过大数据处理，生成可供线上推理的数据吗？就是针对文中大数据出口的第二点。

这是第一节课的课后留言，你可以先回忆一下第一节的内容，然后再听我讲。在推荐服务器做线上推断时，实时用户请求里面包含的特征一般是直接在服务器内部提取出来的，所以肯定不需要再在数据流中走一遍。

但是线上请求数据最终还是会落盘，生成日志数据。这个过程中，一些流处理和批处理的平台会对这些数据做进一步处理，生成今后可供我们使用的特征以及训练用样本。

问题 9：王老师，在线预测的时候，模型所需的特征是直接从数据库读取，还是在线实时组装？我在想如果只是用户或者物品自身的特征的话，可以从数据库读取，但如果是用户和物品的交叉特征的话，是不是必须实时组装？

非常好的点。一般来说如果组合特征可以在线处理，最好能够在线处理，因为组合特征有组合爆炸问题，为了节约宝贵的存储资源，我们一般不直接存储。

但对于一些不得不存储的组合特征，比如用户 x 物品的曝光、点击记录，如果线上模型需要的话，还是要存储到数据库中的，因为这些特征你没办法在线组合。

问题 10：为什么深度学习的结构特点不利于稀疏特征向量的处理呢？

首先，我想说这个问题问得太好了，如果不解决这个问题，那整个 Embedding 技术的意义就没有了，所以我也希望你能好好思考一下这

个问题。

一方面，如果我们深入到神经网络的梯度下降学习过程就会发现，特征过于稀疏会导致整个网络的收敛非常慢，因为每一个样本的学习只有极少数的权重会得到更新，这在样本数量有限的情况下会导致模型不收敛。另一个方面，One-hot 类稀疏特征的维度往往非常地大，可能会达到千万甚至亿的级别，如果直接连接进入深度学习网络，那整个模型的参数数量会非常庞大，这对于一般公司的算力开销都是吃不消的。

所以基于上面两个原因，我们往往先通过 Embedding 把原始稀疏特征稠密化，然后再输入复杂的深度学习网络进行训练，这相当于把原始特征向量跟上层复杂深度学习网络做一个隔离。

好了，这节课就到这里。非常感谢前 8 讲对内容有深度思考和提问的同学，你们的每个问题都很精彩。在接下来的课程中，欢迎你继续畅所欲言，把留言区这个工具好好利用起来，我们一起进步！