推荐系统融合排序的多目标寻优技术

程引 | 快手科技 资深推荐算法工程师

DataFunSummit # 2024

个人简介

知乎专栏链接

微信请简单介绍下自己~

上海交通大学

博士研究生 上海

滴滴出行

派单算法 / 强化学习 北京

BizReach

推荐算法 / 自然语言处理 东京

SmartNews

推荐算法 东京 / 北京

快手科技

推荐算法 北京

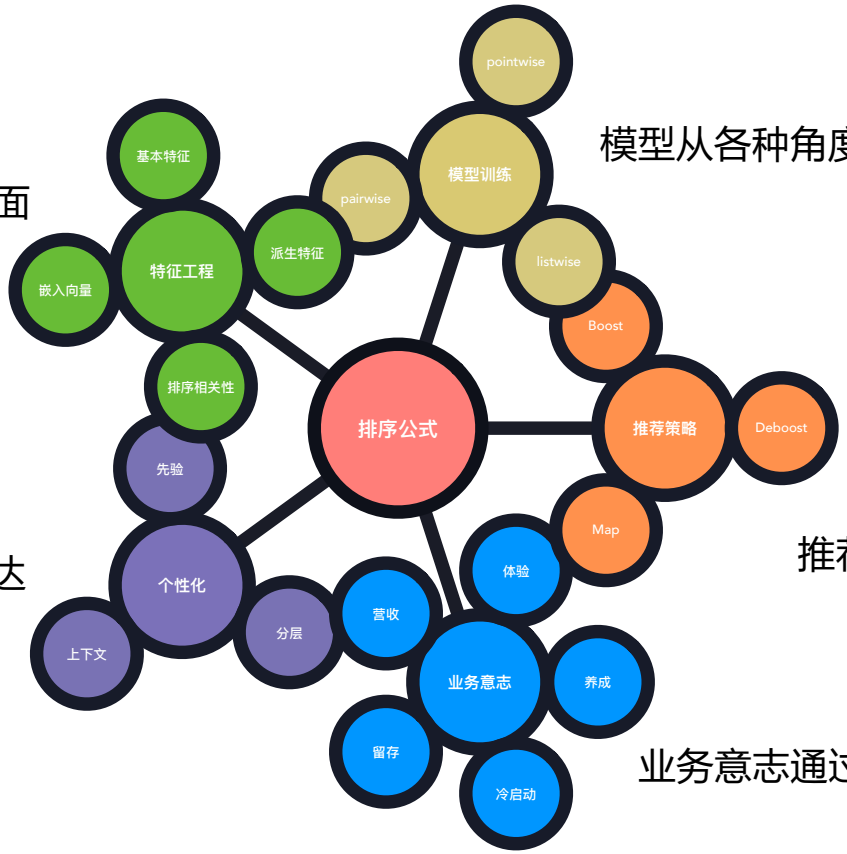
目录
CONTENT

排序公式：推荐系统的核心

推荐系统的最终呈现形式就是排序

特征工程丰富排序公式覆盖面

个性化差异通过排序公式表达



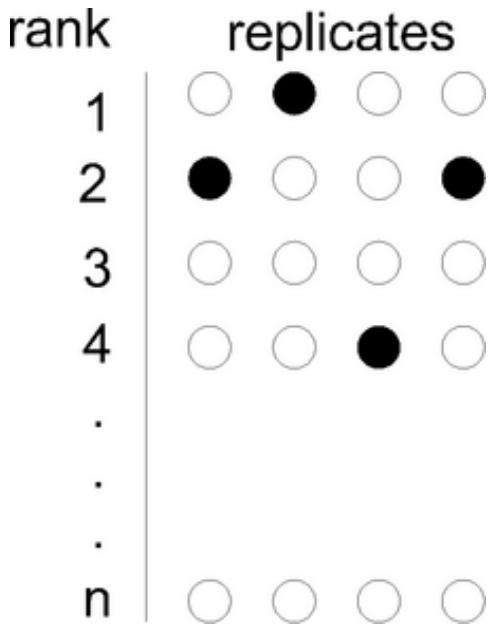
模型从各种角度学习排序因子

推荐策略通过排序公式生效

业务意志通过排序公式嵌入

排序公式的两类融合方式：序融合

序融合关注的是各推荐算法提供的项目或内容的排名，而不直接使用具体的数值。



优点

不受不同推荐算法输出尺度的影响，因为它只关心排名。

更适合处理那些只能输出排名而无法输出精确评分的推荐算法。

缺点

忽略了评分的具体信息，可能会丢失某些推荐系统的敏感度。

对于项的数量和排名的一致性要求较高。

适用条件:

各推荐系统的输出质量差异较大，但都能提供可靠的排序。

需要整合不同类型推荐系统的结果，这些系统可能基于不同的用户交互数据和反馈机制。

排序公式的两类融合方式：值融合

值融合指的是直接在推荐算法的输出值上进行操作。

优点

直观且易于实现，尤其是在各个推荐器输出具有相似尺度和解释性的情况下。

在数据丰富且各来源差异不大的情况下效果较好。

缺点

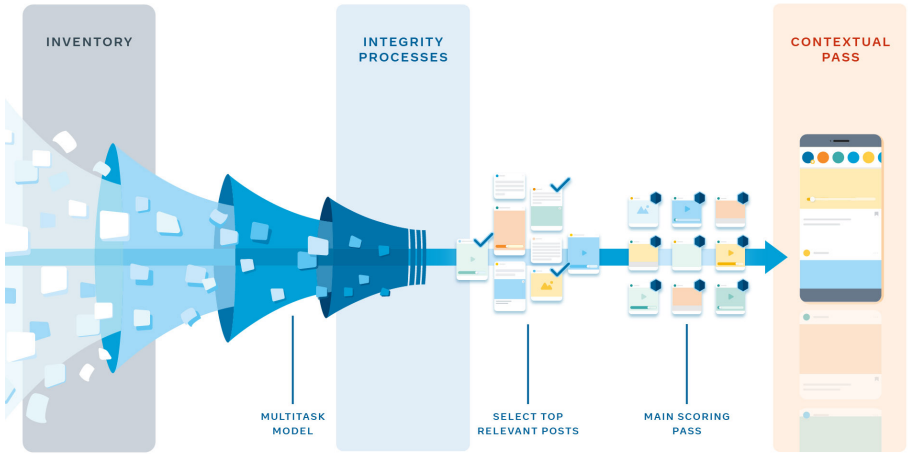
对不同尺度或分布的值需要先进行归一化或标准化。

对异常值较为敏感，特别是采用简单平均等方法时。

适用条件:

各数据来源的可信度相近或已知。

推荐算法的输出为直接可比较的评分或概率值。

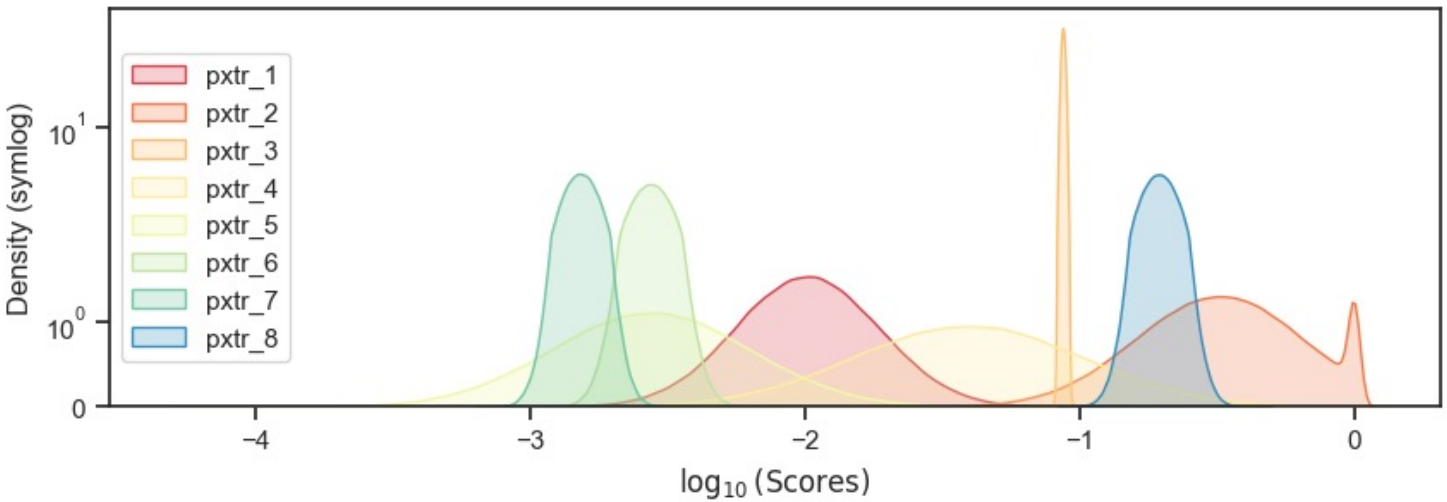


什么是好的融合排序公式

融合公式至少要满足以下几个基本要求：

1. **具有区分度**：区分是排序的核心功能，较弱的区分度意味着排序机制的失效。
2. **尽可能保留排序队列的信息量**：排序队列理应向融合分施加影响，否则就没有实现融合。
3. **尽可能少的冗余超参数**：超参数应该能直接体现业务意志，过多的超参数会让融合公式丧失调整（不论是人工或自动）的指向性与被理解的能力。

扩展阅读：值融合排序公式设计哲学的数学原理



业务夹角现象与多目标权衡

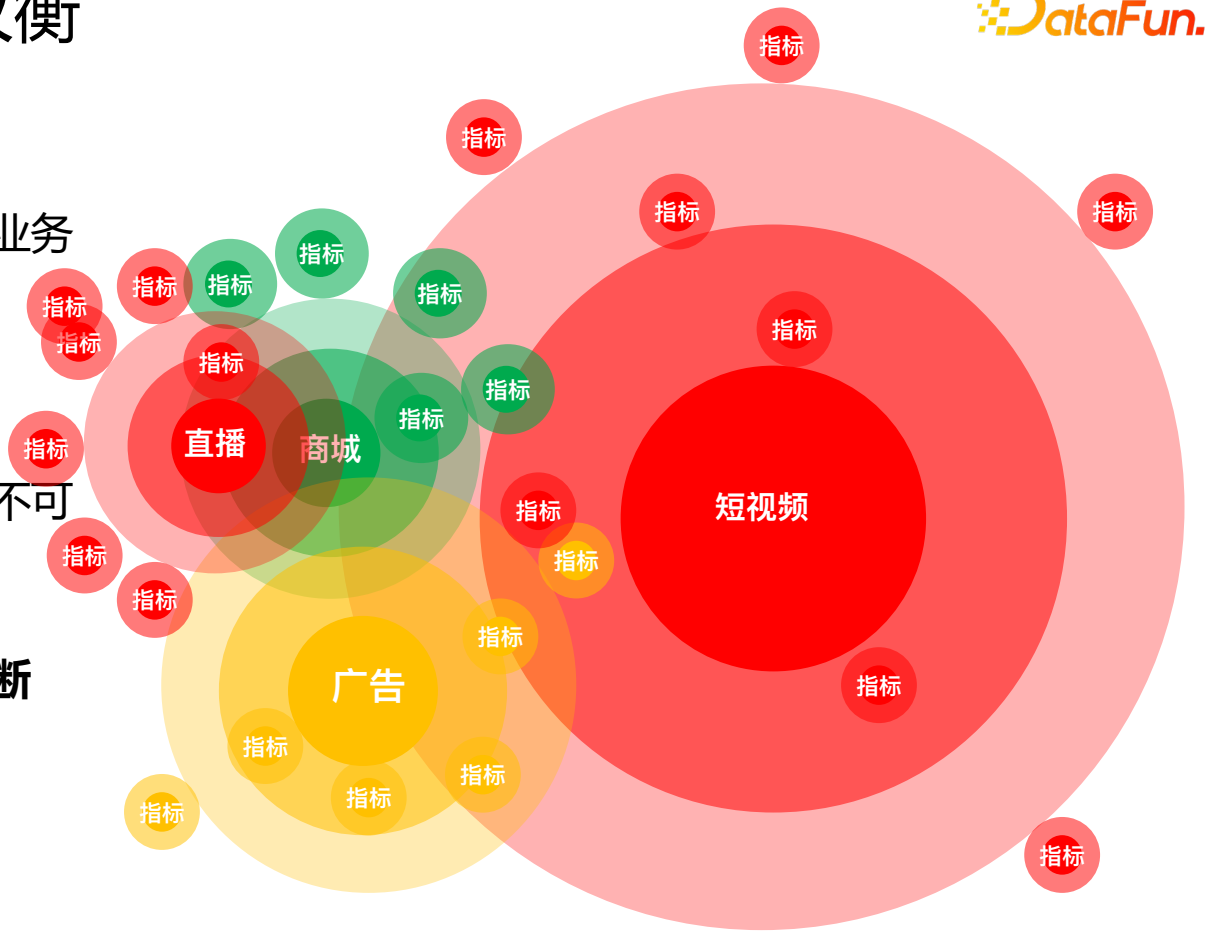
随着版图扩张，一定会产生相互龃龉的业务

每个业务都会有自己的核心/重点指标

帕累托改进在多数的迭代优化中几乎是不可能的

多目标权衡的核心：价值判断与事实判断

价值判断：业务之间的折算比
事实判断：业务置换的难易度



排序公式离线寻参的技术原理

为什么寻参，为什么要离线?

调公式、调参数在各个公司都是效果显著、需求量巨大，而又费时费心费力的工作。

目前多数自动化寻参工具的有效迭代需要生成大量的样本进行评估。这将在实践中带来以下几个困难：

难以保证寻参质量：在线寻参需要大量的样本进行评估，线上小流量寻参无法有效覆盖样本空间。造成寻参质量不稳定，波动大的问题。

难以提升寻参效率：在线寻参算法无法实现快速收敛，白白浪费大量的真实流量迭代生成样本，且难以实现并行化。

难以解多目标优化：实践中我们的对于不同业务指标的优化难度，以及相关取舍是反复调整的过程中才能确定的，事先确定的融合目标一定不是最终我们落地的融合目标。

排序公式离线寻参的技术原理

离线寻参的基本步骤

1. 给定一个融合公式的权重 w，可以计算出一个 $\text{ensemble}_{\text{score}}$

2. 在一个离线样本里，用 $\text{ensemble}_{\text{score}}$ 和用户的行为，如 click, watch, follow, gift 等计算出离线指标，比如 $\text{AUC}_{\text{click}}$，$\text{AUC}_{\text{follow}}$

3. 将各种行为的离线指标融合成一个离线目标，比如：$\text{target} = \text{AUC}_{\text{clic}} + \text{AUC}_{\text{follow}}$

4. 找出使得离线指标最大的权重 $w^* = \arg\max_{w} \ target$

由此可知，离线寻参框架至少需要以下功能：计算融合分、评估子目标、融合子目标、优化总目标。

ParaDance 融合排序量化寻参实战

基于上述讨论，我开发了一套开箱即用的，融合排序公式自动寻参框架： **ParaDance**

开箱即用、离线计算、高速并行、评估手段丰富、支持公式形式多样、子目标可控的多目标优化

PARADANCE

Installation

You can install ParaDance using pip:

```
pip install --upgrade paradance
```

ParaDance 融合排序量化寻参实战

步骤O： 准备离线数据

根据我们的需求，筛选排序可能得相关因子，收集落盘离线数据（此处展示为 fake 样例）。

	pXTR_0	pXTR_1	pXTR_2	pXTR_3	click	share
0	0.272868	0.397742	0.252717	0.758096	0	1
1	0.354293	0.164032	0.102388	0.503173	0	1
2	0.275586	0.217603	0.487439	0.176722	0	0
3	0.370498	0.194613	0.381984	0.832541	0	1
4	0.335603	0.197520	0.658558	0.516685	0	1
...	...	...	...	...	...	...
9995	0.504919	0.658983	0.650045	0.668313	0	1
9996	0.205361	0.189239	0.310282	0.990000	0	1
9997	0.438527	0.358022	0.579011	0.588117	0	1
9998	0.172669	0.345302	0.549418	0.424433	0	0
9999	0.539078	0.526662	0.572708	0.901669	0	1

ParaDance 融合排序量化寻参实战

步骤一：确定融合公式的形式

目前 ParaDance 已经支持:

加法公式（equation_type = 'sum'）：

$$\text{Score} = \text{weights}[0] \times \text{pXTR}_0 + \text{weights}[1] \times \text{pXTR}_1 + \text{weights}[2] \times \text{pXTR}_2 + \text{weights}[3] \times \text{pXTR}_3$$

乘法公式（equation_type = 'product' ）：

$$\text{Score} = \text{pXTR}_0^{\text{weights}[0]} \times \text{pXTR}_1^{\text{weights}[1]} \times \text{pXTR}_2^{\text{weights}[2]} \times \text{pXTR}_3^{\text{weights}[3]}$$

随心所欲公式（equation_type = 'free_style' ）：

$$\text{Score} = 1 + \text{weights}[0] \times \text{columns}[0] + \text{weights}[1] \times \text{columns}[1] + \text{weights}[2] \times \text{columns}[1] \times \text{columns}[2] + \text{weights}[3] \times \text{columns}[3]$$

此外还支持非常复杂的，以 json 形式直接传入的融合公式(equation_type ='json')

ParaDance 融合排序量化寻参实战

步骤一： 确定融合公式的形式， 构造融合公式

```
import paradance as para
```

```
selected_columns=['pXTR_0', 'pXTR_1', 'pXTR_2', 'pXTR_3']
equation_eval_str = '1 + weights[0] * columns[0] + weights[1] * columns[1] + weights[2] * columns[1]* columns[2] + weights[3] * columns[3]'
equation_type = 'free_style'
weights_num=4
```

```
cal = para.Calculator(
    df=df,
    selected_columns=selected_columns,
    equation_type=equation_type, # "sum" or "product" or "free_style or "json"
    equation_eval_str=equation_eval_str,
)
```

在这个例子中，我们选取了自由形式的融合公式，并且涉及到 4 个可以调节的超参数，因此取 weights_num=4

ParaDance 融合排序量化寻参实战

步骤二： 确定子目标，构造多目标优化器

代码中，我们定义了综合目标为第一个子目标的两倍，加上第二个子目标。

两个子目标分为 click 和 share 的 auc。
综合目标的优化方向为「最大化」，同时给出了超参数的上下界。

注意： 除了 auc 算子以外，ParaDance 支持非常丰富的子目标算子。

```
formula = "2*targets[0]+targets[1]"
```

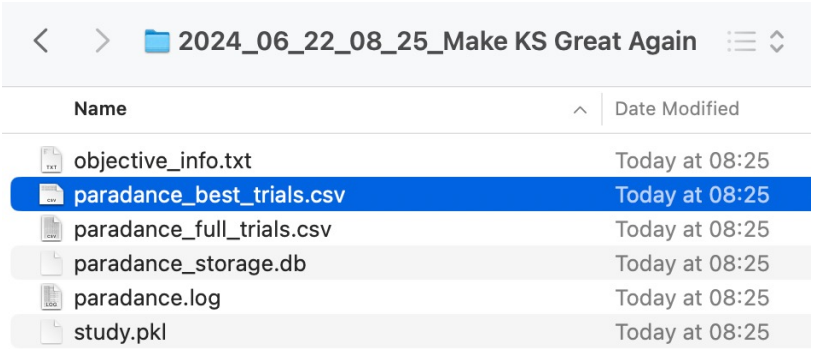
```
ob = para.MultipleObjective(
    calculator=cal,
    direction="maximize",
    formula=formula,
    weights_num=weights_num,
    free_style_lower_bound=[0, -1, 1, -1],
    free_style_upper_bound=[1, 10, 100, 50],
    study_name="Make KS Great Again",
)

ob.add_evaluator(
    flag="auc",
    target_column="click",
)

ob.add_evaluator(
    flag="auc",
    target_column="share",
)
```

ParaDance 融合排序量化寻参实战

步骤三： 自动寻优，看结果

```
para.optimize_run(ob, n_trials=300)
```

运行完成之后，本次实验的优化结果会显示在，以时间戳为前缀，以 study_name 为后缀的文件夹中。
以上介绍了本框架最初级的应用，目前已支持非常复杂的寻优逻辑。



paradance_best_trials

2*targets[0]+targets[1]	Trial	['auc', 'auc']	['pXTR_0', 'pXTR_1', 'pXTR_2', 'pXTR_3']
2.152904173591330	1	[0.6332957777777778, 0.8863126180357755]	[0.9657971630472295, 7.877009035306058, 2.445262734086608, 6.444869670739921]
2.1553320904293900	159	[0.6494476666666666, 0.8564367570960559]	[0.9425410026651977, 3.6734313069004454, 5.86312016718554, 4.069256853840638]
2.161925416305050	141	[0.622538, 0.9168494163050531]	[0.9385581324019349, 4.0396167187023, 1.005965519206514, 4.376628678046488]
2.1649313546572100	184	[0.6360341111111111, 0.8928631324349906]	[0.8707900345839584, 3.20067904747588, 3.644755143605608, 4.066498996249761]
2.165316151628860	189	[0.6361428888888889, 0.8930303738510788]	[0.9595489195218436, 3.357235036241543, 4.078958733510046, 4.403160615515707]
2.167183109410220	205	[0.6628782222222221, 0.8414266649657799]	[0.8912430548702474, 4.752523127238501, 2.8145363756831108, 3.100033154376459]
2.1808087848821200	226	[0.6659533333333333, 0.8489021182154561]	[0.8519907334968821, 4.719704090222412, 1.044434518455862, 2.6705448739731628]
2.2214254424246800	266	[0.6757596666666668, 0.869906109091347]	[0.8083312480302882, 2.127561176264086, 1.0763251138056154, 1.628446015847547]
2.228503021635350	271	[0.6905034444444446, 0.8474961327464621]	[0.8080589420935304, 1.7151438973049435, 1.1803828300684667, 1.207818342396159]

paradance_best_trials记录了历次最佳寻优超参以及对应的子目标和综合目标优化情况。
文件夹中还包含了配置信息、完整的迭代记录和 check_point。
代码样例见右侧二维码，来练练手吧~


感谢观看
DataFun.