第 5 章 粗排模块

在上一章中，我们介绍了推荐系统的召回模块。召回阶段通过多路召回策略从海量内容库中快速筛选出数千到数万个候选物品，从而保证推荐系统具备足够的覆盖能力与探索能力。然而，召回阶段的主要目标是“找得到”，而不是“排得准”。

随着互联网内容规模的持续增长，现代推荐系统每天需要面对数千万甚至上亿级别的内容供给。即使经过召回阶段的初步筛选，候选集合规模通常仍然达到几千甚至上万个。如果直接将如此大规模的候选集送入复杂的精排模型，不仅会带来巨大的计算开销，也难以满足在线服务对于延迟和吞吐的要求。因此，在召回与精排之间，工业界普遍引入了粗排（Pre-Ranking）阶段。粗排的核心任务是在保证较高召回率的前提下，利用轻量级模型对候选集进行进一步筛选，将候选规模从成千上万压缩到数百的量级，从而为后续精排模型腾出计算预算。

与此同时，随着推荐系统从单目标优化逐渐演化为多目标优化，粗排阶段所承担的职责也越来越丰富。它不仅需要完成高效筛选，还需要承担流量调控、候选集优化、多目标融合以及生态约束等重要任务。因此，粗排已经从早期简单的过滤模块逐渐演化为推荐系统中承上启下的关键决策层。本章将重点介绍粗排模块的整体架构与核心技术，包括粗排模型、多目标融合、链路一致性以及粗排后处理等内容。

5.1 为什么需要粗排

从系统架构角度来看，粗排模块位于召回与精排之间，其主要职责是对召回结果进行快速筛选与初步排序。召回阶段的目标是保证内容覆盖率，因此往往会采用多路召回策略，从不同角度为用户寻找潜在感兴趣的内容。虽然这种方式能够有效提升候选集的丰富度，但也会带来大量相关性较弱甚至噪声较高的候选物品。如果直接将这些召回出的候选物品送入精排模型，不仅会增加在线推理成本，还会降低整体系统效率。因此，需要通过粗排阶段对候选集合进行进一步压缩。粗排模型通常采用参数规模较小、计算效率较高的轻量化架构，在较低计算成本下完成候选集的第一轮质量筛选，从而将候选规模压缩至数百级别。

除了候选集压缩之外，粗排还承担着多目标融合的重要职责。现代推荐系统往往需要同时优化点击率（CTR）、观看时长（Watch Time）、点赞率（Like）、评论率（Comment）、转发率（Share）、转化率（CVR）等多个目标，这些目标通常统称为 pxtr（Predictive eXpected Through Rate）。由于不同目标对应的排序结果往往存在差异，而最终呈现给用户的只能是一条排序序列，因此需要通过多目标融合机制将多个目标统一到同一个排序框架之中。

与此同时，推荐系统还需要适应业务目标的动态变化。例如，在产品增长阶段，平台可能更加关注用户时长与活跃度；而在商业化阶段，则可能更加关注广告收入、GMV 或长期用户价值。因此，无论是在粗排还是精排阶段，系统都需要具备灵活的流量调控能力，通过动态调整目标权重和流量分配策略，实现不同业务目标之间的平衡与切换。

除了业务目标之外，推荐系统还需要满足平台生态层面的约束。例如，为保障不同业务、垂类内容以及内容创作者的曝光机会，系统通常会引入 Quota 控制、保量策略、截断机制等手段，从而在推荐效率与生态公平之间取得平衡。从更高层次来看，粗排的核心目标并非精确刻画用户对单个物品的兴趣程度，而是从**候选集合优化（Candidate Set Optimization）**的角度出发，对召回结果进行高效的粗粒度筛选，为后续精排阶段构建一个质量更高的候选集。一个优质的候选集通常需要同时满足以下几个特征：

- 具有较高的相关性，能够覆盖用户潜在感兴趣的内容；
- 具有合理的多样性，避免候选过度集中于单一内容类型；
- 具有一定的探索性，为新内容和长尾内容保留曝光机会；
- 满足平台生态与业务约束要求。

因此，粗排本质上是在有限计算资源约束下，对候选集合进行质量优化与结构优化的过程，其目标是在召回效率、候选质量、多样性以及探索性之间取得平衡，为后续精排阶段保留足够的优化空间。粗排模块的整体结构如图5.1所示。接下来我们将依次介绍粗排模型、链路一致性、多目标融合以及粗排后处理等核心模块。

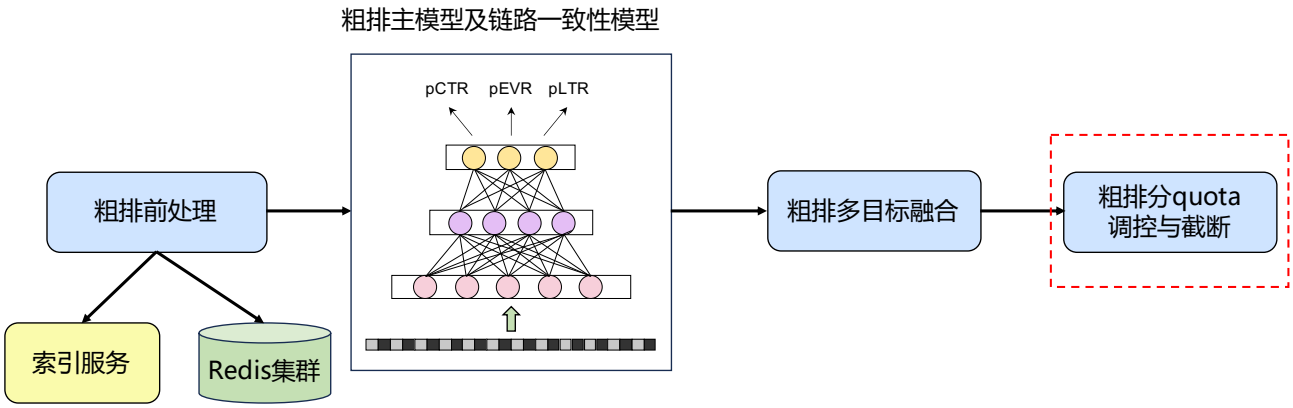

图 5.1: 粗排模块内部处理流程。

5.2 粗排模型

粗排模型（Pre-Ranking Model）是粗排阶段最核心的组成部分，其主要职责是对召回阶段返回的大规模候选集合进行快速评估与筛选。在上一章中我们已经介绍过，多路召回模块的主要目标是提升候选集的覆盖率和召回率。因此，召回阶段更关注“找得到（Retrieve）”，而不强调排序精度。经过多路召回融合后，一个用户请求通常会产生数千甚至上万个候选物品。然而，这些候选物品中既包含用户真正感兴趣的内容，也包含大量相关性较弱甚至噪声较高的内容。如果直接将全部候选物品送入精排模型，不仅会带来巨大的计算开销，也会导致在线服务时延难以满足要求。因此，粗排阶段的首要任务便是对候选集进行进一步压缩。通常情况下，粗排会将候选规模从数千级别压缩至数百级别，使后续精排模型能够将有限的计算资源集中投入到更高质量的候选物品上。

从系统资源分配角度来看，粗排模型实际上承担着推荐链路中的第一次大规模资源分配任务。由于粗排阶段决定了哪些候选物品能够进入后续精排，因此其本质上是在有限计算预算约束下，对候选集进行一次粗粒度筛选（Coarse-Grained Filtering）。与精排模型相比，粗排模型通常面临更加严格的时延约束。工业级推荐系统往往要求单次推荐请求在数百毫秒内完成响应，而粗排阶段能够获得的推理预算通常最多只有 50 毫秒。与此同时，其处理的候选规模却远高于精排阶段。因此，粗排模型必须在预测效果与计算效率之间进行权衡。

早期工业界广泛采用的粗排模型是**双塔模型（Two-Tower Model）**，如图5.2所示。双塔模型由用户塔（User Tower）和物品塔（Item Tower）组成，分别对用户侧特征和物品侧特征进行编码，得到对应的向量表示。最终通过向量相似度计算 (比如向量点积) 获得用户与物品之间的匹配分数。双塔结构最大的优势在于计算效率高。由于用户向量和物品向量可以分别离线计算，因此在线阶段只需要进行向量检索和点积运算即可完成打分。这种架构非常适合粗排场景对于高吞吐、低延迟的要求。

随着推荐系统的发展，粗排模型逐渐从单目标排序演化为多目标排序。现代推荐系统往往需要同时预测点击率（CTR）、观看时长（Watch Time）、点赞率（Like）、评论率（Comment）、转发率（Share）、转化率（CVR）等多个用户行为目标。因此，在实际工程中，粗排模型通常会采用多任务学习（Multi-Task Learning）架构，通过多个任务塔同时输出不同目标对应的 pxtr 预估值，为后续的多目标融合提供基础输入。

然而，粗排模型天然受到算力预算的限制，其表达能力往往远弱于后续精排模型。一方面，用户特征与物品特征在编码阶段往往需要保持较高程度的解耦，以保证在线推理效率，因此难以充分建模复杂的高阶交互关系；另一方面，粗排阶段能够使用的特征数量、模型深度以及网络参数规模均远低于精排阶段。这使得粗排模型虽然具备较好的筛选能力，但其预测精度通常无法达到精排模型的水平。

从某种意义上说，粗排模型并不追求“看得最准”，而是追求“淘汰得足够准”。只要能够以较低成本快速过滤掉大部分低质量候选物品，其目标便已经基本达成。因此，粗排模型的优化目标并非无限逼近最终业务指标，而是在有限计算预算下尽可能提升候选集质量，为后续精排阶段保留更大的优化空间。

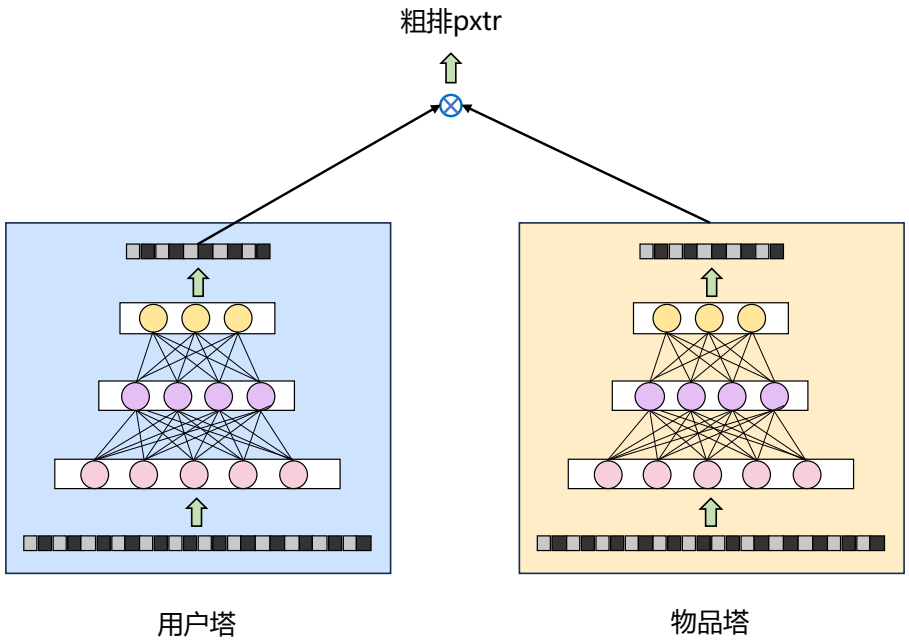

图 5.2: 粗排模型的双塔架构。

5.3 链路一致性模型

虽然粗排模型能够有效完成候选集压缩任务，但级联式推荐架构仍然面临一个长期存在的重要问题：**链路不一致（Pipeline Inconsistency）**。从系统架构角度来看，召回、粗排和精排分别承担不同职责，并且往往由不同团队独立负责优化。由于各阶段采用的特征体系、训练样本、优化目标以及模型结构并不完全相同，因此不同阶段对于“优质内容”的定义往往存在差异。例如，粗排模型可能更加关注点击率，而精排模型则同时关注点击率、观看时长、互动率以及长期留存价值。在这种情况下，一个物品即使能够获得精排模型的高分，也有可能由于粗排阶段的判断偏差而被提前过滤掉。这种现象通常被称为**漏斗效应（Funnel Effect）**。

由于推荐系统采用逐层过滤的级联结构，上游阶段的每一次筛选都会永久丢失一部分候选物品。一旦优质内容在粗排阶段被过滤，即使后续精排模型拥有更强的预测能力，也无法再将其重新召回。因此，上下游模型之间的不一致会直接影响整个推荐链路的效果上限。为了缓解这一问题，工业界逐渐提出了链路一致性（Consistency）优化思想。链路一致性的核心目标并不是提升粗排模型本身的预测精度，而是让粗排模型尽可能学习下游精排模型的排序偏好。

一种最常见的方法是将精排结果作为教师模型（Teacher），利用知识蒸馏（Knowledge Distillation）[5] 或者级联学习（Cascade Learning）的方式训练粗排模型。例如，可以将精排 Top-K 物品构造为正样本，将未进入 Top-K 的候选物品构造为负样本，从而训练粗排模型学习精排阶段的排序决策边界。从本质上看，这种方法相当于将精排模型中蕴含的复杂排序知识向上游传递，使粗排模型在不增加在线推理成本的情况下获得部分精排能力。

由于精排模型通常拥有更丰富的特征体系和更强的表达能力，因此链路一致性模型往往能够在首次上线时获得较为明显的收益：

- 提高粗排与精排之间的排序一致性；
- 减少优质候选物品被提前过滤的概率；
- 提升进入精排阶段候选集的整体质量；
- 在不增加在线时延的情况下复用精排知识。

因此，链路一致性模型长期以来一直是工业界粗排优化的重要方向之一。然而，链路一致性并不是推荐系统优化的最终目标。从本质上看，一致性指标属于系统内部指标（System Metric），而用户时长、用户满意度、互动率以及商业化收益等则属于业务指标（Business Metric）。两者之间虽然存在一定相关性，但并不存在严格的因果关系。

当粗排模型已经能够较好地复现精排偏好后，继续提升一致性指标所带来的收益往往会快速衰减。此时系统可能陷入一种“精排复读机（Ranking Mimicry）”状态：粗排不断模仿精排，而精排本身的能力却没有发生提升。更进一步地说，如果粗排与精排完全一致，那么实际上整个链路就退化成了两个功能相同但复杂度不同的排序器串联运行。此时继续优化一致性已经失去了实际意义。

因此，近年来工业界对于链路一致性的理解也逐渐发生变化。链路一致性不再被视为最终优化目标，而更多被视为一种提升候选集质量、减少漏斗损失的工程手段。真正决定推荐系统上限的，仍然是最终业务目标、多目标优化能力以及用户长期价值建模能力。

5.4 粗排多目标融合

在粗排主模型、链路一致性模型以及其他辅助模型完成推理之后，系统通常会得到多个不同目标对应的预测结果，例如点击率（CTR）、转化率（CVR）、点赞率（LTR）、有效播放率（EVTR）以及链路一致性模型输出的 pxtr 排序分数等。然而，不同 pxtr 所对应的排序结果往往存在明显差异。例如，以点击率为目标排序得到的结果，未必能够带来最高的转化率；而以转化率为目标排序得到的结果，也未必能够最大化用户停留时长。由于推荐系统最终只能向用户输出一条统一的排序序列，因此需要将多个目标进行统一建模，并融合为一个最终排序分数：

$$s_{agg} = f(p_1, p_2, \cdots, p_n) \tag{5.1}$$

其中，p_i 表示不同目标对应的 pxtr 预测值，$f(\cdot)$ 表示多目标融合函数。

从推荐系统的发展历程来看，多目标融合大致经历了四个阶段：

- 基于值（Value-Based）的融合；
- 基于序（Rank-Based）的融合；
- 值序融合（Hybrid Fusion）；
- 自动化融合（Auto Fusion）。

下面分别对这四种多目标融合方式进行介绍。

5.4.1 基于值的融合

基于值（Value-Based）的融合方法直接利用 pxtr 的数值进行计算，是工业界最早也是最常见的一类融合方式。根据融合函数形式的不同，又可以进一步分为线性融合与乘法融合两类。

5.4.1.1 线性融合

工业界最早广泛采用的是线性融合（Linear Fusion）：

$$s_{agg} = w_1 p_{ctr} + w_2 p_{cvr} + \cdots + w_n p_{cascade} \tag{5.2}$$

其中，w_i 表示对应目标的权重，反映该目标对于最终排序结果的重要程度。

线性融合最大的优势在于简单、稳定且易于解释。例如：提高 p_{ctr} 权重，可以增加点击率；提高 p_{cvr} 权重，可以提升转化效率；提高 p_{vtr} 权重，可以增加用户停留时长。因此在线性融合体系下，算法工程师可以直接通过调整权重实现流量调控。

然而，线性融合也存在明显缺陷。首先，它只能表达目标之间的加性关系：$f(a, b) = a + b$；而无法刻画目标之间的协同关系，例如 $p_{ctr} \times p_{ltr}$ 往往比单独的 CTR 或 LTR 更有价值。因此工业界经常引入高阶交叉因子：$p_{ctr}p_{ltr}$、$p_{ctr}p_{cvr}$、$p_{ltr}p_{evtr}$ 并将这些交叉项加入融合公式中。

在实际工程中，这种不断向排序体系中加入新因子的过程通常被称为：

加队列（Add Queue）

其本质是在现有排序体系中引入新的信息来源，从而寻找增量收益。

另一方面，线性融合对于 pxtr 的绝对数值高度敏感。例如 $p_{evtr} \gg p_{cmtr}$ 有效播放率往往远高于评论率。又比如 $CTR_{ModelA} \neq CTR_{ModelB}$，即使两个模型排序能力相近，由于校准程度不同，其输出分布也可能存在较大差异。这会导致原有权重失效，从而增加调参成本。随着排序因子不断增加，权重之间会产生复杂耦合关系，使线性融合逐渐演变成一个难以维护的大型经验公式。

5.4.1.2 乘法融合

为了缓解线性融合容易被单个目标主导的问题，工业界逐渐引入了**乘法融合（Multiplicative Fusion）**。其典型形式如下：

$$s_{agg} = \prod_{i=1}^{n} (\beta_i + \alpha_i p_i)^{\gamma_i} \tag{5.3}$$

其展开形式为：

$$s_{agg} = (\beta_1 + \alpha_1 p_1)^{\gamma_1} (\beta_2 + \alpha_2 p_2)^{\gamma_2} \cdots (\beta_n + \alpha_n p_n)^{\gamma_n} \tag{5.4}$$

与线性融合相比，乘法融合更加关注多个目标之间的协同表现。在线性融合下，一个物品只要某个目标极高，就可能获得较高排序分数。而在乘法融合下：$p_i \to 0$ 会导致整体得分显著下降。因此，乘法融合天然具有短板效应（Bottleneck Effect），更倾向于选择多个目标同时表现优秀的内容。关于加法公式与乘法公式的数学性质、Taylor 展开分析以及协同增益机制，我们将在后续章节进行详细讨论。

5.4.2 基于序关系的融合

虽然基于值的融合方法具有较强的表达能力，但其对 pxtr 的具体数值分布高度敏感。在工业实践中经常会出现这样一种情况：模型升级之后，排序能力实际上有所提升，但由于输出分布发生变化，导致原有融合策略失效，从而使线上 A/B 实验效果持平甚至下降。

为了提高融合策略的鲁棒性，工业界逐渐提出了**基于排序序关系（Rank-Based Fusion）的融合方式**。其核心思想是不再关注具体数值，而只关注排序位置。首先分别对各个目标进行排序：

$$Rank_{ctr}, Rank_{cvr}, \cdots, Rank_n \tag{5.5}$$

然后利用排序位置构造融合分数：

$$s_{agg} = w_1 \frac{1}{Rank_{ctr}} + w_2 \frac{1}{Rank_{cvr}} + \cdots + w_n \frac{1}{Rank_n} \tag{5.6}$$

这种方法最大的优势在于其对于 pxtr 的具体取值范围、分布形态以及校准程度并不敏感，仅依赖于排序顺序本身，因此具有较好的鲁棒性。在实际工业系统中，$\frac{1}{Rank}$ 只是最基础的一种形式，还衍生出了大量变种。例如幂函数形式：

$$S_i = \frac{w_i}{Rank_i^{\lambda} + b} \tag{5.7}$$

归一化幂函数形式：

$$S_i = w_i \left(1 - \left(\frac{Rank_i}{N}\right)^{\lambda}\right) \tag{5.8}$$

指数形式：

$$S_i = w_i \left(e^{k(2\frac{Rank_i}{N} - 1)} - e^{-k(2\frac{Rank_i}{N} - 1)}\right) \tag{5.9}$$

以及带平滑项的排序衰减形式：

$$S_i = w_i \left(1 - \left(\frac{Rank_i}{N+b}\right)^{\lambda}\right) \tag{5.10}$$

其中，N 表示候选集大小，λ、k 和 b 为可调超参数。

不过，纯 Rank 融合也存在明显缺陷。由于只保留了排序位置，而丢失了原始 pxtr 的数值信息，因此无法感知不同物品之间真实的分数差异。例如排名第一和第二的物品可能分数极其接近 ($score_1 = 0.99, score_2 = 0.98$)，也可能存在数量级差异 ($score_1 = 0.99, score_2 = 0.50$)，但在 Rank 融合中都会被视为相邻位置。这种信息的损失实际上也会影响最终的排序效果。

5.4.3 基于值与序关系的融合

为了同时保留数值信息与排序信息，工业界逐渐发展出了值序融合（Hybrid Fusion）方法。其一般形式如下：

$$s_{agg} = f(pxtr, Rank_{pxtr}) \tag{5.11}$$

进一步可以近似表示为：

$$s_{agg} = f(pxtr)g(Rank_{pxtr}) \tag{5.12}$$

其中，$f(\cdot)$ 负责建模 Value 信息，$g(\cdot)$ 负责建模 Rank 信息。例如，$f(pxtr) = \log(1.0+pxtr)$，$f(pxtr) = (\beta+\alpha \cdot pxtr)^{\gamma}$，$g(Rank_{pxtr}) = \frac{1}{Rank^{\lambda}}$ 或者 $g(Rank_{pxtr}) = 1 - \left(\frac{Rank}{N}\right)^{\lambda}$。这样既能够保留模型输出的细粒度数值差异，又能够获得 Rank 融合带来的鲁棒性。目前大量工业级推荐系统都采用了某种形式的值序混合融合策略。

5.4.4 自动化融合与贝叶斯优化

前面介绍的线性融合、乘法融合以及值序融合，本质上都属于**人工设计排序公式（Hand-Crafted Fusion）**的范畴。在这种范式下，算法工程师需要不断调整融合公式中的权重参数，例如：

$$score = w_1 p_{ctr} + w_2 p_{cvr} + w_3 p_{evtr} + \cdots + w_n p_n \tag{5.13}$$

中的 $w_1, \cdots, w_n$ 这 n 个超参数，或者：

$$score = \prod_{i=1}^{n} (\beta_i + \alpha_i p_i)^{\gamma_i} \tag{5.14}$$

中的 $\alpha_1, \cdots, \alpha_n$，$\beta_1, \cdots, \beta_n$，$\gamma_1, \cdots, \gamma_n$ 这 3n 个超参数。随后，算法工程师通过线上 A/B 实验观察业务指标变化，并不断迭代参数配置。

然而，随着排序因子数量不断增加，参数空间会迅速膨胀。例如：10 个排序因子，每个权重有 20 种候选取值；则参数组合数量达到 20^{10}，此时依赖人工经验进行调参已经变得十分困难。因此，工业界开始尝试利用自动化搜索算法完成融合参数优化，其中最具代表性的方案便是**贝叶斯优化（Bayesian Optimization）** [4]。

从推荐系统角度来看，多目标融合参数优化本质上是一个黑盒优化（Black-Box Optimization）问题。假设融合参数表示为：

$$\mathbf{w} = (w_1, w_2, \cdots, w_n) \tag{5.15}$$

对于任意一组参数，可以计算融合分数：

$$score = f(\mathbf{w}, pxtr) \tag{5.16}$$

随后，我们可以利用离线样本进行评估，例如可以计算 AUC_{click}、AUC_{follow} 等用户反馈标签对应的离线指标。然后，我们进一步可以构造综合目标函数，即：

$$Target = AUC_{click} + AUC_{follow} + \cdots \tag{5.17}$$

或者：

$$Target = \lambda_1 AUC_{click} + \lambda_2 AUC_{follow} + \cdots \tag{5.18}$$

最终目标变为：

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} Target(\mathbf{w}) \tag{5.19}$$

这便构成了一个典型的参数优化问题。然而这里的目标函数往往具有无法获得解析表达式、无法计算梯度、可能非凸、不可微、每次评估成本较高等特点。因此，传统基于梯度下降的优化方法难以直接应用。贝叶斯优化专门用于解决如下形式的问题：

$$x^* = \arg\max_{x \in \mathcal{A}} f(x) \tag{5.20}$$

其中 $f(x)$ 为未知目标函数，$\mathcal{A}$ 为参数搜索空间，每次评估 $f(x)$ 的代价较高。

贝叶斯优化的核心思想可以概括为：

定义 5.1

利用一个简单模型去近似复杂目标函数，并利用该模型指导后续搜索。

♣

因此，贝叶斯优化通常由两个核心组件组成，即**代理模型（Surrogate Model）**和**采集函数（Acquisition Function）**。代理模型负责对未知目标函数进行建模。工业界最经典的代理模型是高斯过程（Gaussian Process，GP）[3, 8, 11]。假设已经观测到：

$$\mathcal{D} = (x_i, f(x_i))_{i=1}^n \tag{5.21}$$

高斯过程假设：

$$f(x) \sim GP(\mu_0(x), K(x, x')) \tag{5.22}$$

即任意有限个点上的函数值都服从联合高斯分布。对于新的参数点 x，根据贝叶斯后验推断，可以得到：

$$f(x)|\mathcal{D} \sim N(\mu_n(x), \sigma_n^2(x)) \tag{5.23}$$

其中 $\mu_n(x)$ 表示预测收益，$\sigma_n(x)$ 表示预测不确定性。此时代理模型不仅能够预测当前参数组合可能获得的收益，还能够评估预测结果是否可靠。这也是贝叶斯优化区别于传统回归模型的重要特点。

仅有代理模型还不足以完成优化。因为我们真正关心的是：

下一次应该尝试哪个参数组合？

这由采集函数（Acquisition Function）决定。采集函数需要在两种目标之间进行平衡：Exploration（探索）与 Exploitation（利用）。利用意味着优先尝试当前看起来最优的参数区域，而探索意味着优先尝试不确定性较高的未知区域。

工业界最常用的采集函数之一是期望增益（Expected Improvement，EI）。定义当前最优值：

$$f_n^* = \max_{i \le n} f(x_i) \tag{5.24}$$

则 EI 定义为：

$$EI(x) = \mathbb{E}[\max(f(x) - f_n^*, 0)] \tag{5.25}$$

其含义是选择最有可能超过当前最优结果的参数点。因此下一次采样点为：

$$x_{n+1} = \arg\max_x EI(x) \tag{5.26}$$

理论上，贝叶斯优化最经典的实现方式是：

$$GP + EI \tag{5.27}$$

即高斯过程作为代理模型，EI 作为采集函数。

然而，在工业级推荐系统中，参数维度往往较高。例如：

$$\mathbf{w} = (w_1, w_2, \cdots, w_{50}) \tag{5.28}$$

此时高斯过程的计算复杂度会迅速增长。因此，很多工业系统会采用 Tree-structured Parzen Estimator（TPE）[10] 代替 GP。TPE 同样属于贝叶斯优化框架，但其思想不再直接建模 $P(y|x)$ 而是建模 $P(x|y)$，从而能够更好地处理高维参数空间。目前广泛使用的超参数优化框架 Optuna 默认采用的便是 TPE 算法。因此在推荐系统实际工程中，很多所谓的“贝叶斯优化寻参”，其底层实现往往并非 GP，而是基于 TPE 的贝叶斯优化框架。

在实际推荐系统中，贝叶斯优化通常作为参数更新模块的一部分。例如：排序参数、融合队列参数、乘法融合超参数、通道配额超参数等。推荐系统会周期性收集历史实验结果 $(\mathbf{w}_i, reward_i)$ 并利用贝叶斯优化算法搜索新的参数组合。随后将搜索得到的新参数配置下发到线上系统进行验证。整体流程如下图5.3所示：

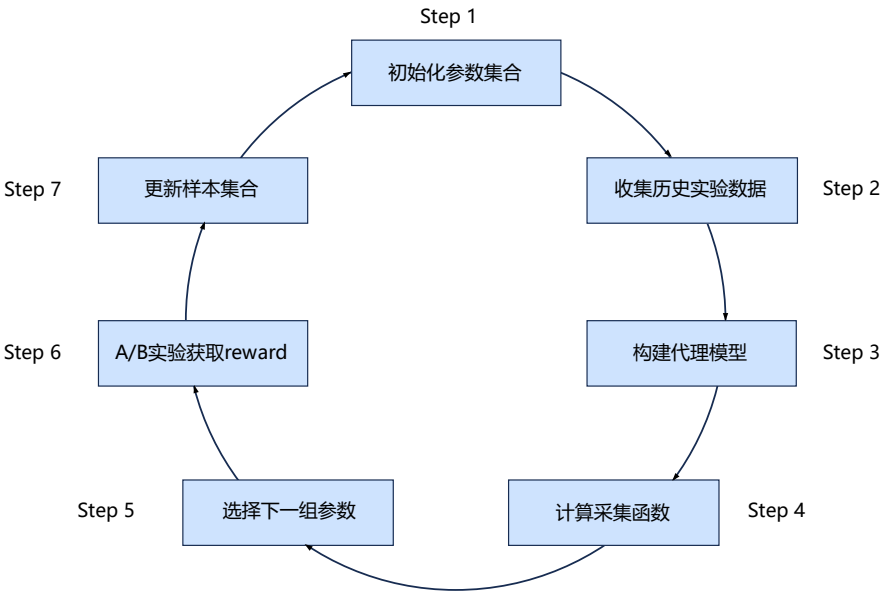

图 5.3: 推荐系统中贝叶斯优化具体流程。

从本质上看，贝叶斯优化解决的是：

笔记 如何利用尽可能少的实验次数找到尽可能优的排序参数。

因此，贝叶斯优化已经成为现代推荐系统策略优化与多目标融合中的重要基础工具之一。除了贝叶斯优化方法之外，近几年工业界推荐系统中也逐渐发展出了 Cross-Entropy Method（CEM）、Learning-to-Fuse、End-to-End Ranking、强化学习排序等自动化排序技术。这些方法不再关注单纯的参数搜索，而是直接搜索排序策略本身。相关内容将在下一章精排模块以及第24章“多任务融合”中进一步展开讨论。

5.5 多目标融合机制分析

在前一节中，我们介绍了粗排多目标融合任务下常见的几种融合方式。其中，应用最广泛的仍然是加法公式（Additive Fusion）与乘法公式（Multiplicative Fusion）两种形式。虽然两类公式都能够实现多个目标的融合排序，但它们对于流量分配的偏好却存在本质差异。不同的融合方式不仅会影响最终的排序结果，还会进一步影响平台流量的分布形态以及头部内容的组成结构。因此，理解加法公式与乘法公式背后的排序机制，对于设计合理的多目标优化策略具有重要意义。

为了便于分析，假设当前仅有点击率（CTR）与转化率（CVR）两个目标参与融合，则加法公式可以表示为：

$$score = w_1 p_{ctr} + w_2 p_{cvr} \tag{5.29}$$

最简单的乘法公式则可以表示为：

$$score = p_{ctr}^{\alpha} p_{cvr}^{\beta} \tag{5.30}$$

考虑表5.1中的三个候选 Item。假设 $w_1 = w_2 = 1$，则会有 $score_A = score_B = score_C = 1.0$。在加法公式下，A、B、C 三个 item 的分数完全相同。而在乘法公式下，$score_A = 0.09$，$score_B = 0.25$，$score_C = 0.09$。可

以发现 B 这个 item 的分数是最大的。

表 5.1: 多目标融合排序 ctr、cvr 示例。

Item	p_{ctr}	p_{cvr}
A	0.9	0.1
B	0.5	0.5
C	0.1	0.9

从上述例子可以观察到一个非常重要的现象：加法公式与乘法公式对于不同类型内容的偏好并不相同。在加法融合中，只要某一个目标表现足够突出，即使其他目标表现一般，仍然有机会获得较高的最终得分。因此，加法公式更容易将某些在单一目标上表现优异的内容推向排序头部。而在乘法融合中，任意一个目标过低都会显著拉低最终得分。因此，只有多个目标同时表现较好的内容，才能获得较高的排序位置。为了便于理解，我们可以将二者类比为不同的人才选拔机制：

- 加法公式更偏好“偏科生”，允许某个维度特别突出；
- 乘法公式更偏好“六边形战士”，要求多个维度均衡发展。

不过，这一结论仅适用于最简单的两个因子直接相乘的乘法公式形式。后续我们将看到，在工业界广泛使用的广义乘法公式中，这种“强双好”特性会随着偏置项 β 的引入而逐渐减弱。

我们仍以 p_{ctr} 和 p_{cvr} 这两个因子的融合为例，下图5.4是分别使用加法公式和乘法公式时的等值线。当 $p_{ctr}+p_{cvr}=1$ 时，在加法公式下是图5.4中左图的红色虚线，而在乘法公式中则 $p_{ctr}p_{cvr}=1$ 只能退化成一个取值为 $(1.0,1.0)$ 的孤点。除此之外，当 $p_{ctr}+p_{cvr}=c$ 或者 $p_{ctr}p_{cvr}=c$ 公式中 $c=0.5$ 时，在加法公式中 p_{ctr} 和 p_{cvr} 取值最大值为 0.5，反之在乘法公式中 p_{ctr} 和 p_{cvr} 的最小值为 0.5，且另外一个值只能取 1.0。从两个图的黑色线可以看出，左图加法公式两个因子取值的数值远小于乘法公式下两个因子的取值。因此可以看出在达到相同 c 的目标下，乘法公式对于融合因子的取值大小要求是更高的。

同时，我们也可以从 $(0.9,0.1)$ 与 $(0.99,0.01)$ 这两个点在加法公式下与乘法公式下的取值 c 进行对比。在加法公式下两个点在一条等值线上，而乘法公式下 $(0.99,0.01)$ 所在的等值线更加接近 0。这说明乘法公式对于单个因子的极端取值具有很强的“惩罚效应”，这种效应也被称为**短板效应（Bottleneck Effect）**。

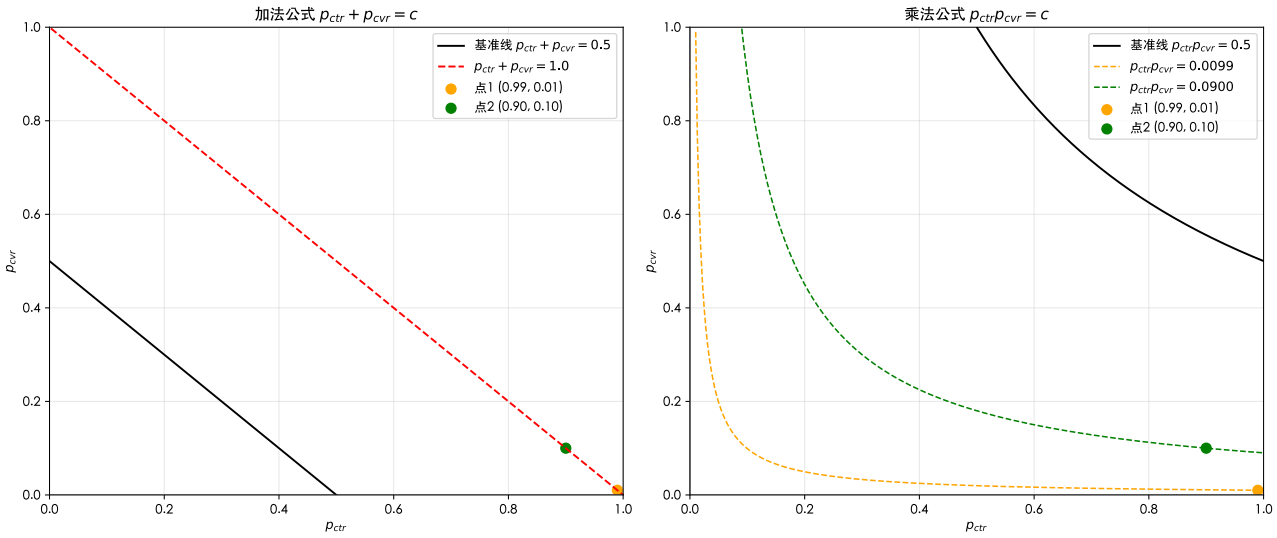

图 5.4: 加法公式与乘法公式等值线对比。

除此之外，乘法公式可以看做是对数空间的加法公式，由于对 pxtr 的取值做了对数变换，而 $pxtr\in[0,1]$，这样做对数变换之后，天然会显著放大 $[0,1]$ 区间内的取值差异。这其实也能解释，在电商领域，经常需要将 CTR 和 CVR 作为乘法的因子组合在一起，而不是加法公式组合。因为这样从乘法公式性质的角度去看，电商业务需要 CTR 和 CVR 都高的物品，而不需要高 CTR 低 CVR 或者高 CVR 低 CTR 的物品。

总结起来，加法公式融合与乘法公式融合的性质如下：

定义 5.2

加法融合是一种补偿型（Compensatory）排序机制，高目标可以补偿低目标；而乘法融合是一种短板约束型（Bottleneck-Constrained）排序机制，任何一个目标过低都会显著拉低最终得分。

在前面的讨论中，我们使用的乘法公式主要是 $p_{ctr}p_{cvr}$ 这种因子直接相乘的形式。而在工业界推荐系统中，通常会使用前面小节中用到的通用乘法公式形式，即 $\prod_{i=1}^{n}(\beta_i+\alpha_i p_i)^{\gamma_i}$。

下面可以分几种情况来分析。第一种情况是 $\beta=0$，这个时候没有偏置项，整个乘法公式退化成了因子相乘的形式，只不过单个因子为 $s_i=(\alpha_i p_i)^{\gamma_i}$，这时候还是与前面讨论一样，乘法公式具有选择“强双好”的特性。而随着偏置项 β 的增大，整个乘法公式会逐渐趋于加法公式，如下图5.5所示。当 $\beta=0$ 和 $\beta=1$ 时，整个乘法公式的曲线仍然是双曲线，而当 $\beta=10$ 时，整个乘法公式的双曲线已经非常接近直线了，即加法公式的线性融合形式，而对于 $(0.9,0.1)$ 与 $(0.5,0.5)$ 的因子取值，在 $\beta=0$ 和 $\beta=1$ 时乘法公式的取值差异很大，而当 $\beta=10$ 时，乘法公式的取值为 110.09 和 110.25，几乎没有什么差别。

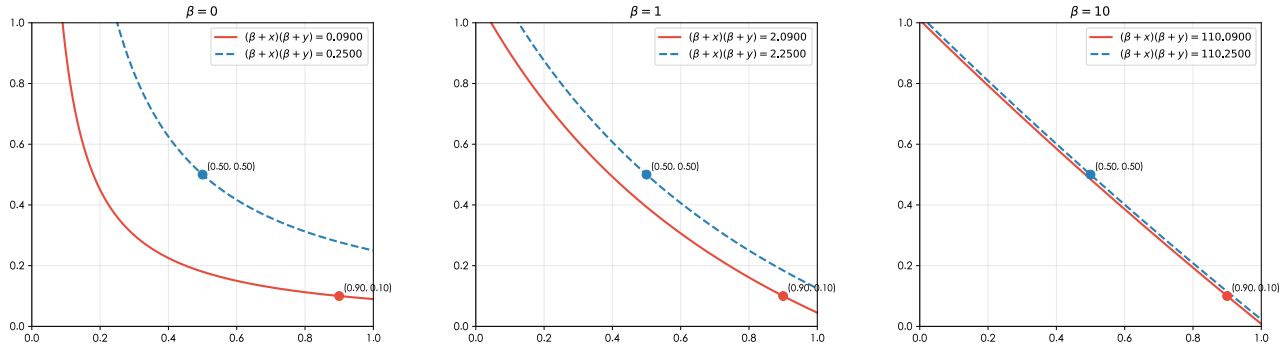

图 5.5: 乘法公式中偏置项 β_i 对于等值线的影响。

前面的分析主要基于几何视角与数值示例展开。虽然能够帮助我们直观理解加法公式与乘法公式的差异，但仍然无法解释一个更加本质的问题：

为什么工业界广泛采用的广义乘法公式，在某些情况下会表现出类似加法公式的行为?

为了回答这一问题，下面从数学 Taylor 展开的角度进一步分析广义乘法公式的结构。

$$Score=\prod_{i=1}^{n}(\beta_i+\alpha_i p_i) \tag{5.31}$$

将 β_i 提出：

$$Score=\prod_{i=1}^{n}\beta_i\prod_{i=1}^{n}\left(1+\frac{\alpha_i p_i}{\beta_i}\right) \tag{5.32}$$

定义：

$$Const=\prod_{i=1}^{n}\beta_i \tag{5.33}$$

以及：

$$x_i=\frac{\alpha_i p_i}{\beta_i} \tag{5.34}$$

则有：

$$Score=Const\prod_{i=1}^{n}(1+x_i) \tag{5.35}$$

当 $n=3$ 时：

$$\begin{aligned}\prod_{i=1}^{3}(1+x_i) =&(1+x_1)(1+x_2)(1+x_3)\\ =&1+(x_1+x_2+x_3)\\ &+(x_1x_2+x_1x_3+x_2x_3)\\ &+x_1x_2x_3\end{aligned} \tag{5.36}$$

进一步代入 x_i 可得：

$$Score = Const\left[1+\sum_i \frac{\alpha_i p_i}{\beta_i}+\sum_{i<j}\frac{\alpha_i\alpha_j p_i p_j}{\beta_i\beta_j}+\sum_{i<j<k}\frac{\alpha_i\alpha_j\alpha_k p_i p_j p_k}{\beta_i\beta_j\beta_k}+\cdots\right] \tag{5.37}$$

其中：

- 一阶项对应线性加权求和；
- 二阶项对应目标之间的两两协同作用；
- 三阶项对应三个目标同时优秀时的协同奖励；
- 更高阶项对应更复杂的多目标交互关系。

当 $\beta_i \gg \alpha_i p_i$ 时，

$$x_i = \frac{\alpha_i p_i}{\beta_i} \ll 1 \tag{5.38}$$

根据等价无穷小原理：

$$x_i x_j = o(x_i) \tag{5.39}$$

$$x_i x_j x_k = o(x_i) \tag{5.40}$$

即 $x_i x_j \to 0$、$x_i x_j x_k \to 0$。因此可以忽略二阶及以上高阶项：

$$Score \approx Const\left(1+\sum_i \frac{\alpha_i p_i}{\beta_i}\right) \tag{5.41}$$

即：

$$Score \approx Const + \sum_i w_i p_i \tag{5.42}$$

其中：

$$w_i = Const \cdot \frac{\alpha_i}{\beta_i} \tag{5.43}$$

可以发现，此时乘法公式逐渐退化为加法公式。因此，$\beta_i \to \infty$ 时，乘法融合逐渐接近线性加法融合；而 $\beta_i \to 0$ 时，高阶交互项占据主导地位，模型更强调多目标同时优秀的内容。

上述推导揭示了一个非常重要的事实：

乘法公式并不是一种完全不同于加法公式的融合机制。

从数学结构上看，广义乘法公式实际上可以分解为：

$$\text{线性加法项} + \textbf{高阶交互项} \tag{5.44}$$

其中：

- 一阶项决定了各目标的基础贡献；

- 二阶项描述两个目标之间的协同增益；
- 三阶项描述三个目标同时优秀时的额外奖励；
- 更高阶项则进一步刻画复杂的多目标协同关系。

因此，乘法公式真正引入的并不是简单的“相乘操作”，而是目标之间的协同机制（Synergy Mechanism）。高阶交互项越强，模型越倾向于奖励多个目标同时优秀的内容；高阶交互项越弱，模型越接近传统的线性加权求和。

接下来，我们介绍一下幂指数 γ 这个超参数的影响。考虑 $Score = (\beta + \alpha p)^\gamma$，取对数可得：

$$\log score = \gamma \log(\beta + \alpha p) \tag{5.45}$$

当 $\gamma > 1$ 时，高分样本进一步被放大。例如，$0.8^2 = 0.64$，$0.2^2 = 0.04$，二者差距显著增大。因此，我们可以得出以下结论，即：

定义 5.3

在多目标融合机制的策略算法中，乘法公式较大的 γ 会强化头部内容的竞争优势，使流量进一步向高质量内容集中。

♣

而当 $0 < \gamma < 1$ 时，$0.8^{0.5} = 0.894$，$0.2^{0.5} = 0.447$，二者差距被缩小。此时流量分布会更加平缓。因此 γ 本质上控制着流量分布的**陡峭程度（Traffic Sharpness）**。

最后，我们从梯度的视角去看加法公式与乘法公式。考虑两个目标：

$$Score = (\beta_1 + \alpha_1 p_1)(\beta_2 + \alpha_2 p_2) \tag{5.46}$$

对 p_1 求偏导可以发现：

$$\frac{\partial Score}{\partial p_1} \propto (\beta_2 + \alpha_2 p_2) \tag{5.47}$$

即 p_1 的重要性依赖于 p_2 的大小。当 p_2 较大时，$\frac{\partial Score}{\partial p_1}$ 也会增大。这说明：

一个目标越优秀，另一个目标的提升价值越高。

这体现了目标之间的正向协同（Positive Synergy）。而对于加法融合：

$$Score = w_1 p_1 + w_2 p_2 \tag{5.48}$$

有 $\frac{\partial Score}{\partial p_1} = w_1$，$\frac{\partial Score}{\partial p_2} = w_2$，梯度始终为常数。因此各目标之间彼此独立，不存在协同增益。

综上所述，加法融合与乘法融合并非简单的公式形式差异，而是代表了两种完全不同的排序机制。加法融合本质上是一种**补偿型（Compensatory）排序机制**。不同目标之间彼此独立，高目标可以补偿低目标，因此更容易产生“偏科型内容”。乘法融合本质上是一种**协同增强型（Synergistic）排序机制**。目标之间通过高阶交互项产生耦合关系，一个目标的提升价值会受到其他目标状态的影响，因此更倾向于选择多个目标同时优秀的内容。

从工业实践角度来看，广义乘法公式实际上构造了一条从纯乘法到线性加法的连续过渡路径：

- 当 β 较小时，高阶交互项占主导地位，模型强调多目标协同；
- 当 β 较大时，高阶交互项逐渐消失，模型逐步退化为线性加法；
- γ 则进一步控制流量分布的陡峭程度以及头部内容的竞争强度。

因此，工业界广泛采用的广义乘法公式，本质上是在加法融合与协同增强之间寻找一种可调节的平衡机制，从而实现用户体验、商业价值与内容生态之间的动态权衡。

5.6 粗排分 Quota 调控与截断

在完成粗排模型打分、多目标融合以及链路一致性排序之后，系统通常已经获得了一批按照综合分数排序的候选物品。然而，粗排阶段的工作并未结束。由于推荐系统不仅需要优化用户体验，还需要兼顾内容生态、商业化目标以及平台长期增长，因此仅依赖排序分数进行筛选往往是不够的。在工业界推荐系统中，粗排阶段通常

还需要承担候选集调控（Candidate Set Control）的职责。其中最重要的一类技术便是 Quota 调控（Quota Control）、保量（Guarantee）以及截断（Truncation）。

5.6.1 粗排 Quota 调控

首先我们来回答一个问题，即：

笔记 粗排模块为什么需要 Quota 调控机制？

假设在一个直播推荐系统中，当前有如下几种候选池：游戏直播池、电商直播池、秀场直播池、新主播池、高价值主播池。如果完全按照排序分数筛选 $TopK(score)$，则最终候选集可能全部来自同一个候选池。

例如：

表 5.2: 无 Quota 约束下的候选集示例。

主播 ID	类别	Score
A	游戏	0.98
B	游戏	0.96
C	游戏	0.95
D	游戏	0.94
E	游戏	0.93

虽然表5.2中的结果在短期业务指标上可能达到最优，但从长期来看往往会带来一系列问题，例如内容同质化、新主播无法获得曝光、商业化内容流量不足以及平台生态失衡等。尤其是在直播推荐场景中，由于直播内容供给远少于短视频内容供给，如果推荐系统长期过度偏好某一类内容垂类或内容形态，流量将逐渐向少数头部主播集中，从而削弱其他主播的开播积极性，进而导致内容供给持续萎缩，最终影响整个平台生态的健康发展。

对于短视频推荐而言，虽然内容供给规模通常远高于直播业务，但如果推荐系统长期向某一类内容倾斜，例如特定时长分段 (duration) 的视频（长视频或短视频）或者某些热门垂类内容（影视、综艺、短剧等），同样会逐渐改变平台内容生态结构。长期来看，这不仅会影响创作者的内容生产方向，还可能改变用户对于平台内容的认知与消费习惯，从而影响用户体验和平台的长期增长。

因此，在工业级推荐系统中，通常会在粗排阶段引入 Quota 机制，对不同内容类型、内容垂类以及流量池进行配额控制，以保证候选集具备足够的多样性，避免推荐系统过度偏向某一类内容，如下表5.3所示。此时粗排阶段需要在保证排序质量的同时满足 Quota 约束。

表 5.3: 示例 Quota 配置。

候选池	配额
游戏直播池	40%
娱乐直播池	30%
电商直播池	20%
新主播池	10%

下面我们给出一个粗排阶段 Quota 调控的示例。

假设粗排最终需要输出 100 个候选。Quota 配置如下：

$$Quota_{game} = 40 \tag{5.49}$$

$$Quota_{entertainment} = 30 \tag{5.50}$$

$$Quota_{ecommerce} = 20 \tag{5.51}$$

$$Quota_{new} = 10 \tag{5.52}$$

一个简单的实现方式如下：

代码 5.1: 粗排分 quota 调控机制伪代码。

```
# 多路召回候选集
candidates = retrieve()
# 粗排模型预测多个pxtr
for item in candidates:
    item.pctr = ctr_model.predict(item)
    item.pcvr = cvr_model.predict(item)
    item.pltr = ltr_model.predict(item)
# 多目标融合
for item in candidates:
    item.score = 0.5 * item.pctr + 0.3 * item.pcvr + 0.2 * item.pltr

# 按业务规则构造不同流量池
game_pool = [x for x in candidates if x.category == "game"]
ent_pool = [x for x in candidates if x.category == "entertainment"]
ecom_pool = [x for x in candidates if x.category == "ecommerce"]
new_anchor_pool = [x for x in candidates if x.is_new_anchor]

# 每个Pool内部排序
game_pool.sort(key=lambda x:x.score, reverse=True)
ent_pool.sort(key=lambda x:x.score, reverse=True)
ecom_pool.sort(key=lambda x:x.score, reverse=True)
new_anchor_pool.sort(key=lambda x:x.score, reverse=True)

# Quota控制
result = []
result.extend(game_pool[:40])
result.extend(ent_pool[:30])
result.extend(ecom_pool[:20])
result.extend(new_anchor_pool[:10])
# 最终候选集
candidate_set = result
```

最终候选集为:

$$CandidateSet = \bigcup_i TopK_i \tag{5.53}$$

从而保证不同类型内容都能够进入后续精排阶段。

前面这种固定 Quota 的策略机制实现起来非常简单，但存在灵活性较差的问题。例如对于重度游戏用户 $P(Game|User) = 0.9$，此时如果继续使用固定 40% 的游戏 Quota 显然是不合理的。因此，工业界推荐系统中也会引入按流量比例的动态 Quota 调控策略:

$$Quota_i = f(User, Context, BusinessGoal) \tag{5.54}$$

例如:

$$Quota_{game} = 50\% \tag{5.55}$$

$$Quota_{ecommerce} = 10\% \tag{5.56}$$

从而实现个性化流量调控。

进一步地，一些系统会利用强化学习或在线优化方法学习最优 Quota 分配策略：

$$\pi(s) \rightarrow Quota \tag{5.57}$$

根据实时业务指标动态调整流量结构。

5.6.2 粗排保量机制

除了 Quota 之外，推荐系统通常还需要实现保量（Guarantee）。例如：

- 新主播保量；
- 新视频保量；
- 广告保量；
- 商业化内容保量。

一种简单实现方式如下：

$$Exposure_i < Target_i \tag{5.58}$$

则：

$$Score_i = Score_i + \Delta \tag{5.59}$$

其中 Δ 表示保量补偿项。另一种方式则是在 Quota 层面直接保留固定曝光位置。例如在粗排最终输出的候选集 $Top100$ 中预留比如 5% 的流量给冷启动的新内容。

5.6.3 粗排截断机制

完成 Quota 和保量之后，系统通常还需要进行截断（Truncation）。假设召回阶段返回 5000 个候选。粗排阶段输出 300 ~ 500 个候选。则需要执行：

$$TopK(CandidateSet) \tag{5.60}$$

的截断操作，例如 K=300。截断的核心目的包括：控制精排计算成本，提高后续模型吞吐，降低推荐系统整体的延迟等。由于精排模型通常远比粗排复杂，因此合理的截断策略对于整体推荐系统性能具有重要影响。

总体来看，Quota、保量以及截断共同构成了粗排阶段最重要的后处理模块。它们保证了候选集不仅具备较高相关性，同时满足业务约束和生态约束，为后续精排阶段提供更加合理的输入。

5.7 粗排候选集集合优化

从系统视角来看，粗排阶段真正优化的对象并不是单个物品（Item），而是整个**候选集合（Candidate Set）**。传统排序模型关注的问题通常是：

$$P(click \mid user, item) \tag{5.61}$$

即预测用户是否会喜欢某一个具体物品。然而对于粗排阶段而言，其核心职责并不是给出最终排序结果，而是在海量候选集中快速筛选出一批高质量候选，为后续精排阶段提供足够优质的搜索空间。

因此，粗排阶段更关注的问题实际上是：

$$P(Value \mid CandidateSet) \tag{5.62}$$

即整个候选集合是否能够满足后续排序阶段对于相关性、多样性、探索性以及业务约束等多方面要求。从这个角度来看，粗排优化本质上属于一个**候选集优化（Candidate Set Optimization）**问题，而非传统意义上的单物品排序问题。

候选集首先需要保证对用户兴趣的充分覆盖（Interest Coverage）。假设某用户近期兴趣分布如下：

- 篮球内容：40%

- 足球内容：30%
- 游戏内容：20%
- 科技内容：10%

如果粗排候选集全部来自篮球内容，那么虽然这些内容都具有较高的预测点击率，但实际上只覆盖了用户兴趣空间的一部分，大量潜在兴趣方向被提前过滤。因此候选集通常需要满足：

$$Coverage(Candidate\ Set) \rightarrow \max \tag{5.63}$$

从而保证用户多个兴趣方向都能够进入后续排序阶段。工业界常见做法包括：多路粗排队列、Quota 配额控制、类目保量策略、兴趣聚类簇的覆盖度约束等。这些方法的本质都是在提升候选集对于用户兴趣空间的覆盖能力。

除此之外，实际工业系统中经常会出现如下情况：

Top10 内容全部来自影视解说、电影切片或者短剧内容。

虽然这些内容的预测分数都很高，但候选集内部高度同质化。这种现象通常被称为：Candidate Concentration，即候选集聚集度过高。为了避免候选集过度集中于某一类内容，工业界通常会引入多样性约束，例如：

$$Similarity(C_i, C_j) < \tau \tag{5.64}$$

或者：

$$Count(Category_i) \leq K \tag{5.65}$$

其中 $Similarity$ 表示内容相似度，τ 表示允许的最大相似度阈值，K 表示某类内容允许进入候选集的最大数量。通过控制候选集聚集度，可以有效避免后续排序阶段陷入局部最优。

在粗排模块中，如果整个模块始终按照预测分数选取：

$$Candidate\ Set = TopK(score) \tag{5.66}$$

这种贪心利用的方式，则新内容、新作者以及长尾内容几乎无法获得曝光机会。长期下去会导致创作者生态固化、新内容冷启动失败、用户兴趣探索能力下降、推荐系统陷入负反馈循环等。因此，在工业级推荐系统中，我们通常会向候选集中注入一定比例的探索流量，即：

$$Candidate\ Set = (1-\epsilon)TopK + \epsilon Explore \tag{5.67}$$

其中 $\epsilon = 5\% \sim 20\%$，探索内容通常来自新内容、新作者、长尾内容等。探索机制本质上是在短期收益与长期收益之间进行平衡。

如果从粗排模型的角度进行考虑，传统粗排模型大多采用 Point-wise 建模方式。对于样本 (u, i)：

$$L_{point} = CrossEntropy(y, \hat{y}) \tag{5.68}$$

这时模型仅关注当前物品是否会被点击。这种方式包含了以下的隐含假设，即：

笔记 pointwise loss 认为每个样本之间都是彼此独立的，因此可以分开进行 loss 计算与建模。

然而在真实推荐场景中，候选集内部物品并不是独立存在的。例如两个内容高度相似的视频同时出现，多个同类型主播同时进入候选集，一个内容进入候选集后会影响其他内容的曝光机会。因此，越来越多的工业级推荐系统开始引入 Context-Aware 信息到粗排模型中，从候选集列表（List）或者集合 (Set) 的角度进行粗排模型建模。

在 Loss 层面，我们可以采用 List-wise 优化思想。假设候选集为：

$$L = \{i_1, i_2, \cdots, i_n\} \tag{5.69}$$

则优化目标变为

$$P(L|u) \tag{5.70}$$

而不是

$$\prod_i P(i|u) \tag{5.71}$$

此时，模型直接学习整个候选列表对于用户价值的贡献。典型的 List-wise 优化目标包括：ListNet [2]、ListMLE [12]、LambdaRank [1]、Soft-NDCG [7, 9] 等。从优化目标上看，$Item \rightarrow List$ 意味着模型开始关注候选集整体质量，而非单个物品质量。虽然 List-wise 已经能够从列表维度进行优化，但其仍然存在一个问题：

候选集本质上是一个集合（Set），而不是序列（Sequence）。

在粗排阶段：$\{i_1, i_2, i_3\}$ 与：$\{i_3, i_2, i_1\}$ 实际上表示的是同一个候选集合。因此，近年来越来越多的研究开始采用 Set-wise 建模方式。其目标变为：

$$P(Value|Set) \tag{5.72}$$

即直接评估整个候选集合的质量。这类方法通常要求模型满足：

$$f(x_1, x_2, \cdots, x_n) = f(x_{\pi(1)}, x_{\pi(2)}, \cdots, x_{\pi(n)}) \tag{5.73}$$

其中 π 表示任意排列。换句话说模型对于输入顺序不敏感。

最新的集合建模方法，比如 Set Transformer[6]、Deep Sets [13] 等都属于这一方向。这些方法能够显式建模候选集内部物品之间的相互关系，例如：内容覆盖度、内容聚集度、类目分布、duration 分布、兴趣覆盖情况、探索内容比例等信息。因此相比传统 Point-wise 模型，其更符合粗排阶段优化候选集的本质目标。需要说明的是，本章主要介绍粗排阶段的系统设计思想。关于 Set Transformer、集合注意力机制以及基于集合建模的粗排优化方法，将在第11章中进行更加详细的介绍。

由于粗排关注的是候选集质量，因此其评价指标也与精排存在明显差异。常见指标包括：

$$Recall@K = \frac{|Relevant \cap CandidateSet|}{|Relevant|} \tag{5.74}$$

用于衡量后续精排偏好的内容是否被粗排成功保留。除此之外，还包括多样性、聚集度、探索率、有效兴趣数、用户偏好垂类数量等指标。这些指标共同衡量候选集是否具备足够的优化空间。因此，从更高层视角来看，我们可以得出如下结论，即：

定义 5.4

精排优化的是物品排序 (Item Ranking)；粗排优化的是候选集合 (Candidate Set)。 ♣

粗排阶段的核心目标并不是找到最终最优内容，而是在有限计算资源约束下构建一个兼具相关性、多样性、探索性以及业务约束的高质量候选集合。只有当候选集本身足够优秀时，后续精排模型才有机会从中筛选出真正满足用户需求的内容。因此，大家在学习粗排模块以及从事粗排模块优化的时候，都要从集合优化的角度来思考如何让粗排模块选出一个更优质的候选集合。

5.8 本章小结

本章我们详细介绍了级联式推荐系统架构中的粗排模块及其核心设计思想。首先，我们从“为什么需要粗排模块”这一问题出发，分析了在海量候选集和严格时延约束下，推荐系统为何需要在召回与精排之间引入粗排阶段。粗排的核心职责并非精确预测用户对单个物品的兴趣，而是在有限计算资源约束下，对召回结果进行快速筛选，为后续精排阶段构建高质量候选集。

随后，我们从粗排模块的内部结构出发，依次介绍了粗排主模型、链路一致性模型、多目标融合以及粗排后处理等关键环节。其中，粗排主模型负责对候选物品进行快速打分；链路一致性模型通过学习下游精排模型的排序偏好，提高上下游链路的一致性；多目标融合模块则将多个不同目标对应的 pxtr 统一映射到同一排序空间；而粗排后处理模块则进一步完成 Quota 调控、保量策略以及候选集截断等工程化处理。

在粗排多目标融合部分，我们系统讨论了工业界常见的多目标融合方法，包括值融合、序融合、值序融合以及自动化融合等不同技术路线。同时，我们从数学推导的角度分析了加法融合与乘法融合两种最常见的融合方式，并通过泰勒展开证明了广义乘法公式本质上可以视为在线性融合基础上引入多阶目标交互项的结果，从而

解释了乘法融合更倾向于选择多目标均衡内容的原因。此外，我们还介绍了基于贝叶斯优化的自动化参数搜索方法，用于替代传统依赖人工经验的权重调节过程。

在粗排后处理部分，我们进一步讨论了 Quota 调控、保量策略以及候选集截断等工程实践问题。通过对不同内容池、业务池和流量池进行配额控制，推荐系统能够在用户体验、商业化目标以及内容生态之间取得平衡，从而避免候选集过度集中于某一类内容。

最后，我们从更高层的系统视角重新审视了粗排模块的优化目标。与精排阶段关注单个物品排序不同，粗排阶段真正优化的对象是**整个候选集合（Candidate Set）**。围绕这一目标，我们介绍了覆盖度优化、多样性优化、探索性优化等候选集优化策略，并讨论了从 Point-wise 到 List-wise，再到 Set-wise 的建模演进趋势。与此同时，我们也简要介绍了 Set Transformer、Deep Sets 等集合建模方法，为后续章节中粗排模型的深入讨论奠定基础。

总体而言，粗排模块承担着连接召回与精排的重要桥梁作用，其本质是在有限算力预算下构建一个兼具相关性、多样性、探索性以及业务约束的高质量候选集合。只有当粗排阶段能够提供足够优质的候选集时，后续精排模型才有机会从中筛选出真正满足用户需求的内容。下一章我们将继续介绍级联式推荐系统架构中的精排模块，并重点讨论精排阶段与粗排阶段在建模目标、模型结构以及优化方法上的差异与联系。

5.9 参考文献

[1] Christopher JC Burges. “From ranknet to lambdarank to lambdamart: An overview”. In: *Learning* 11.23-581 (2010), p. 81.

[2] Zhe Cao et al. “Learning to rank: from pairwise approach to listwise approach”. In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 129–136.

[3] Mark Ebden. “Gaussian processes: A quick introduction”. In: *arXiv preprint arXiv:1505.02965* (2015).

[4] Peter I Frazier. “A tutorial on Bayesian optimization”. In: *arXiv preprint arXiv:1807.02811* (2018).

[5] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the knowledge in a neural network”. In: *arXiv preprint arXiv:1503.02531* (2015).

[6] Juho Lee et al. “Set transformer: A framework for attention-based permutation-invariant neural networks”. In: *International conference on machine learning*. PMLR. 2019, pp. 3744–3753.

[7] Tao Qin, Tie-Yan Liu, and Hang Li. “A general approximation framework for direct optimization of information retrieval measures”. In: *Information retrieval* 13.4 (2010), pp. 375–397.

[8] Carl Edward Rasmussen. “Gaussian processes in machine learning”. In: *Summer school on machine learning*. Springer, 2003, pp. 63–71.

[9] Michael Taylor et al. “Softrank: optimizing non-smooth rank metrics”. In: *Proceedings of the 2008 international conference on web search and data mining*. 2008, pp. 77–86.

[10] Shuhei Watanabe. “Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance”. In: *arXiv preprint arXiv:2304.11127* (2023).

[11] Christopher Williams and Carl Rasmussen. “Gaussian processes for regression”. In: *Advances in neural information processing systems* 8 (1995).

[12] Fen Xia et al. “Listwise approach to learning to rank: theory and algorithm”. In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 1192–1199.

[13] Manzil Zaheer et al. “Deep sets”. In: *Advances in neural information processing systems* 30 (2017).