第 6 章 精排模块

在上一章中，我们介绍了级联式推荐系统中的粗排模块。粗排阶段的核心任务是在有限的计算资源约束下，从召回返回的大规模候选集中快速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（Candidate Set）。从某种意义上说，粗排优化的是候选集合本身，而并非最终的排序结果。

然而，一个高质量候选集并不意味着最终推荐结果一定优秀。候选集中的物品仍然需要进一步评估其与当前用户兴趣之间的匹配程度，并综合考虑用户价值、内容价值以及商业价值等多个目标。因此，在粗排之后，推荐系统通常会引入精排（Ranking）阶段，对候选集合进行更加精细化的建模与排序。

与粗排阶段强调效率优先不同，精排阶段更加关注排序精度（Ranking Precision）。工业界大量先进的用户兴趣建模方法、多任务学习框架、序列建模技术以及特征交叉网络，往往首先落地于精排模块。因此，精排阶段通常也是整个推荐系统中算法迭代最频繁、模型复杂度最高、业务收益最直接的核心模块。

6.1 精排模块架构

在推荐系统的级联式架构中，精排（Ranking）模块负责对粗排阶段输出的候选物品集合进行更加精细化的排序与筛选，并最终选出 Top-K 个物品送入后续的重排模块。相比于粗排阶段通常采用的轻量化模型，精排模型往往拥有更复杂的网络结构、更丰富的特征体系以及更强的非线性表达能力，因此能够对用户兴趣进行更加准确的刻画，从而获得更高的排序精度。

从功能定位上来看，粗排阶段的主要目标是在海量候选集中快速筛选出用户可能感兴趣的候选集合，而精排阶段则是在这一候选集合内部进一步挖掘用户真实偏好，对每个候选物品进行更加精准的价值评估。换句话说：粗排模块关注候选集和的质量，而精排模块关注最终的排序质量。

因此，精排模型通常围绕用户满意度以及业务目标进行建模，例如点击（Click）、观看时长（Watch Time）、点赞（Like）、评论（Comment）、转发（Share）、关注（Follow）、转化（Conversion）以及商业化收益等行为。通过对这些行为进行预测，系统能够更加全面地评估候选物品对于用户和平台的综合价值。

正如第二章介绍级联式推荐系统整体架构时所述，如图2.2所示，精排模块通常接收粗排阶段筛选出的数百个候选物品（通常为 100～500 个），并进一步从中选出约 50～100 个高质量候选物品送入后续重排阶段。在实际工业场景中，精排阶段处理的候选规模并非固定不变，而是与系统延迟预算、在线算力资源以及业务需求密切相关。例如，在高并发场景下，为了保证整体吞吐量，精排阶段可能仅处理 100 个左右的候选物品；而在算力资源较为充足的场景下，则可能扩大候选规模，以提升最终排序效果。

虽然精排阶段处理的候选数量远少于粗排阶段，但由于其直接决定最终推荐结果的主体质量，因此往往承担着推荐系统中最核心的排序优化任务。事实上，工业界大多数推荐算法创新——包括多任务学习（Multi-task Learning）、序列建模（Sequential Modeling）、用户兴趣演化建模（Interest Evolution）、多目标优化（Multi-objective Optimization）以及生成式排序（Generative Ranking）等方向——都主要集中在精排阶段进行研究与落地。

精排模块的整体处理流程如图6.1所示，其核心可以划分为三个阶段：

1. 精排主模型与 Learning to Rank（LTR）模型；
2. 多目标融合（Multi-objective Fusion）；
3. 流量调控与业务约束（Traffic Control）。

其中，精排主模型与 LTR 模型负责对用户与候选物品之间的匹配程度进行建模，并输出各种用户行为反馈以及业务目标对应的预估值，例如点击率（CTR）、观看时长（Watch Time）、点赞率（LTR）、评论率（CMTR）、转化率（CVR）以及其他业务 Reward 等。这些预估值从不同维度刻画了用户对于候选物品的潜在兴趣和价值，是后续排序决策的重要基础。

在获得多个行为目标对应的 pxtr 预估值之后，系统进入多目标融合阶段。由于推荐系统往往同时面临用户体验、内容生态以及商业化收益等多个优化目标，因此需要通过融合公式或者融合模型，将多个 pxtr 与业务目

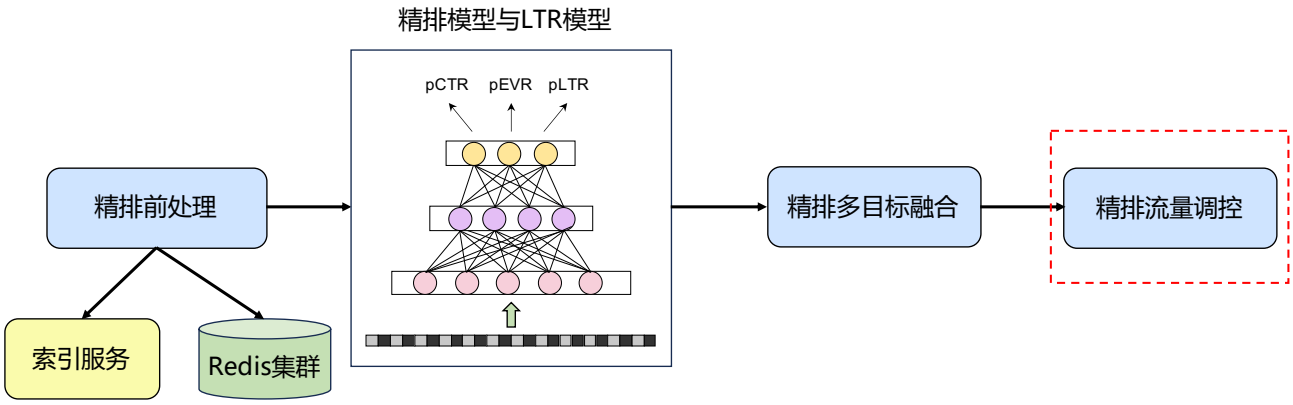


图 6.1: 精排模块内部处理流程。

标统一映射到同一个排序空间中，并最终得到综合排序分数（Ranking Score）。随后精排模块依据融合分数完成候选物品排序，从而形成精排阶段的初步排序结果。

然而，仅仅依赖模型分数进行排序通常难以满足复杂的业务需求。与粗排阶段类似，精排阶段同样需要引入各种业务约束和生态约束，因此通常还会进行流量调控（Traffic Control）。例如：在短视频场景中，需要为新内容提供冷启动流量扶持；在直播场景中，需要保证新主播和长尾主播获得基础曝光；在电商场景中，需要对营销活动、重点商品以及商业化内容进行流量倾斜。除此之外，精排阶段通常还会引入保量（Guarantee）机制，对特定内容池、创作者群体或者业务方向设置最低曝光约束，以避免模型排序导致流量过度向头部内容集中。经过流量调控和保量策略处理之后，精排模块最终从候选集中选出约 50～100 个物品，并将其送入后续重排模块进行进一步优化。

总体而言，精排模块的核心职责是在粗排提供的高质量候选集基础上，尽可能准确地评估用户对于每个候选物品的真实兴趣与潜在价值，并在兼顾业务目标、生态目标以及商业目标的前提下，输出一个兼具效果与合理性的排序结果。如果说粗排模块解决的是“给用户哪些内容”的问题，那么精排模块解决的则是“这些内容应该以什么顺序呈现给用户”的问题。因此，精排模块通常也是整个推荐系统中最核心、最复杂、最具算法创新空间的模块。

6.2 精排模型与 LTR 模型

由于精排阶段处理的候选物品数量远少于粗排阶段，因此其对于单个候选物品的计算资源预算更加充足。基于这一特点，工业界通常采用表达能力更强的深度神经网络（Deep Neural Network，DNN）作为精排主模型的基础架构。与粗排阶段广泛使用的双塔模型不同，精排模型通常会将用户侧特征、物品侧特征以及上下文特征进行统一编码，并拼接（Concatenate）后输入到 DNN 网络中进行建模。由于模型能够同时观察用户与物品两侧的信息，因此可以在网络内部自动学习更加复杂的特征交叉关系和非线性关联。例如，用户年龄与内容类别、用户历史兴趣与当前内容属性之间的高阶交互关系，都可以通过 DNN 结构进行有效建模。

相比于双塔结构，DNN 模型不仅具备更强的特征交互能力，也拥有更高的模型表达能力，因此在排序精度上通常能够取得更好的效果。正因如此，DNN 已逐渐成为工业界精排模型的标准架构之一。需要指出的是，粗排采用双塔架构而精排采用 DNN 架构，并非因为两者在建模能力上存在本质差异，而是推荐系统在效果、算力和时延约束之间进行工程权衡的结果。粗排阶段需要在数千个候选物品上进行快速推理，因此更加关注计算效率；而精排阶段候选集规模已经大幅缩减，可以接受更加复杂的模型结构，从而获得更高的排序精度。从理论上来说，如果系统拥有充足的算力资源并能够满足在线时延要求，那么精排阶段常用的 DNN 架构同样可以迁移到粗排阶段使用。因此，理解粗排与精排模型架构差异背后的工程约束，比单纯记忆模型形式更加重要。

在模型功能层面，精排阶段通常同时包含精排主模型（Main Ranking Model）和 Learning to Rank（LTR）模型两类模型。其中，精排主模型主要负责对用户各种行为反馈进行直接建模。由于推荐系统往往需要同时优化多个目标，因此精排主模型通常采用多任务学习（Multi-Task Learning，MTL）框架，对点击、有效播放、点赞、

评论、收藏、关注、转发等多种用户行为进行联合预估，并输出对应的 pxtr 结果。

而 LTR 模型则更多地从排序目标的角度进行建模。与直接预测用户行为标签不同，LTR 模型通常会基于业务经验构造排序 Reward 或排序监督信号，例如用户综合满意度、长周期价值、高价值用户行为组合等，通过排序学习的方式直接优化最终排序效果。从某种意义上来说，LTR 模型可以看作是对精排主模型预估目标的重要补充，它能够学习一些难以通过单一行为标签刻画的排序偏好。

因此，在实际工业系统中，精排主模型与 LTR 模型往往并非相互替代的关系，而是相互协作、相互补充的关系。前者负责刻画用户具体行为发生的概率，后者负责学习更加贴近最终业务目标的排序偏好，两者共同构成精排阶段最核心的预测与排序能力。

6.3 精排模型常见的预估任务

在介绍精排模型结构之前，我们首先需要了解精排主模型以及 LTR 模型究竟在预估什么。事实上，无论模型结构如何演进，其核心任务始终是对用户未来可能产生的行为以及业务价值进行预测。因此，精排阶段的模型通常包含大量不同的预估任务（Prediction Task），这些任务共同构成了后续排序决策和多目标融合的基础。

从机器学习建模的角度来看，精排阶段常见的预估任务可以大致划分为两类：分类任务（Classification Task）和回归任务（Regression Task）。分类任务主要用于预测某种用户行为是否会发生，因此通常采用离散的 0-1 标签进行建模。例如点击（Click）、点赞（Like）、评论（Comment）、收藏（Favorite）、关注（Follow）、转发（Share）、购买（Purchase）等用户反馈行为，本质上都可以被视为二分类问题。对于某一次曝光而言，如果用户发生了对应行为，则标签记为 1；否则记为 0。模型经过训练后即可输出对应行为发生的概率，即工业界常见的各种 pxtr 指标，例如：

$$p_{ctr}, \quad p_{ltr}, \quad p_{cmtr}, \quad p_{ftr}, \quad p_{cvr} \tag{6.1}$$

这些预估值分别对应点击率、点赞率、评论率、收藏率以及转化率等不同目标。

除了用户的显式互动行为之外，一些隐式反馈也经常采用分类方式进行建模。例如短视频场景中的有效播放率（Effective View Rate）、完整播放率（Finish Rate）、长时观看率（Long View Rate）等指标，本质上都是将连续观看行为转换为某种阈值事件进行预测。例如：

$$label = \begin{cases} 1, & watch\_time > 5s \\ 0, & otherwise \end{cases} \tag{6.2}$$

通过这种方式，模型能够学习用户是否会达到某个预先设定的满意度标准。

另一类常见任务是回归任务。与分类任务不同，回归任务直接预测连续数值，因此更加适用于观看时长、停留时长、消费金额、GMV 等连续型业务指标。例如在短视频推荐场景中，用户观看时长（Watch Time）通常是一个连续变量，因此可以直接采用回归模型进行建模：

$$\hat{y} = f(x) \tag{6.3}$$

其中 $\hat{y}$ 表示模型预测的观看时长。类似地，在电商推荐场景中，也可以直接预测用户未来可能产生的消费金额、订单价值或者生命周期价值（Lifetime Value, LTV）等连续型指标。

除了上述单一行为目标之外，随着推荐系统的发展，越来越多的精排模型开始直接预测业务 Reward。例如用户满意度（Satisfaction）、用户长期价值（Long-Term Value）、商业价值（Revenue）、GMV、广告收益等高层目标。此时模型的预测对象已经不再局限于单个用户行为，而是对多个行为进行加权组合后形成的综合收益指标。

因此，从工业界的角度来看，精排模型和 LTR 模型的预估目标已经逐渐从早期单一的 CTR 预测，演进为覆盖点击、互动、观看、转化、消费以及长期价值等多个维度的多任务学习体系。后续的多目标融合阶段则会进一步将这些不同维度的预估结果统一整合，从而得到最终用于排序的综合分数。

6.3.1 时长预估

时长预估（Watch Time Prediction）是精排模型中最重要的预估任务之一。在短视频、直播以及长视频推荐场景中，用户观看时长往往比点击行为更加能够反映用户的真实满意度，因此时长类指标已经逐渐成为现代推荐系统最核心的优化目标之一。

从机器学习的角度来看，用户播放时长本质上是一个连续值，因此天然属于回归问题。然而在工业实践中，直接对播放时长进行回归建模通常会面临以下几个问题：

- 时长分布呈现严重长尾特征，大量样本集中于较小的时长区间；
- 用户行为噪声较大，相同内容在不同曝光场景下可能产生较大波动；
- MSE 等回归损失容易受到异常样本影响，导致训练不稳定；
- 回归任务通常比分类任务更难收敛。

因此，工业界通常不会直接对原始时长进行回归，而是采用一些特殊的建模技巧，将连续时长目标转换为分类问题或者半分类问题进行学习。

6.3.2 稀疏互动信号建模

除了上一章节介绍的时长预估任务及其分类、回归建模方法之外，精排模型还需要对大量互动类行为信号进行建模，例如点击、点赞、评论、收藏、关注、转发、分享、打赏以及购买等行为。这些互动信号通常对应用户更强的兴趣表达，因此在推荐系统的排序决策中往往具有较高的价值。

然而，与时长信号相比，互动类信号具有一个显著特点，即其数据分布通常非常稀疏。对于时长类目标而言，只要用户对某个曝光物品产生观看行为，系统就能够获得对应的观看时长反馈。因此从数据分布的角度来看，时长信号通常属于相对稠密（Dense）的监督信号。例如在短视频、直播等场景中，大部分曝光都会产生一定的播放时长，因此时长标签的覆盖率通常较高。

而互动类目标则完全不同。以推荐系统中常见的行为指标为例：

- CTR（Click Through Rate）：通常在 $3\% \sim 10\%$ 左右；
- LTR（Like Through Rate）：通常低于 1%；
- CMTR（Comment Through Rate）：往往低于千分级；
- FTR（Follow Through Rate）：通常低于万分级；
- CVR（Conversion Rate）：在电商与广告场景中通常更加稀疏。

随着用户行为价值的不断提升，其对应标签往往也会越来越稀疏。这意味着在训练过程中，绝大部分样本都是负样本，而真正有价值的正样本数量极其有限。因此，在精排模型中，一个非常重要的研究方向便是如何有效学习这些稀疏互动目标。

6.3.3 样本加权

针对稀疏目标，最简单也是最常见的方法是样本加权（Sample Weighting）。其核心思想是提高高价值正样本在训练过程中的梯度贡献，从而缓解正负样本极度不平衡的问题。例如在直播推荐场景中，如果需要建模用户打赏概率（GTR, Gift Through Rate），那么每一次打赏行为往往伴随着具体的打赏金额（G-Value, Gift Value）。此时可以将打赏金额作为正样本权重：

$$w_i = gift\_amount_i \tag{6.4}$$

，对应的加权交叉熵损失为：

$$L = -\sum_i w_i \left(y_i \log p_i + (1 - y_i) \log(1 - p_i)\right) \tag{6.5}$$

这样一来，高金额打赏行为将产生更大的梯度贡献，使模型更加关注高价值用户行为。从业务角度来看，这种方式不仅能够缓解样本稀疏问题，同时也能够引导模型学习不同正样本之间的价值差异。例如打赏 100 元与打赏 1 元虽然都属于正样本，但显然对应着完全不同的用户价值和兴趣强度。

6.3.4 稠密目标辅助稀疏目标学习

除了样本加权之外，另一类非常重要的方法是利用稠密目标辅助稀疏目标学习（Dense-to-Sparse Learning）。其基本思想来源于用户行为之间天然存在的漏斗关系（User Behavior Funnel）。例如在电商推荐场景中：

$$Exposure \rightarrow Click \rightarrow Purchase \tag{6.6}$$

用户必须先点击商品，才有可能产生购买行为。因此 CTR 相比于 CVR 更加稠密，而 CVR 又依赖于 CTR 的发生。基于这种行为依赖关系，研究人员提出了大量经典模型来利用稠密目标辅助稀疏目标学习，其中最具代表性的工作包括：

- CTCVR（Click-Through and Conversion Rate）
- ESMM（Entire Space Multi-Task Model）[3]
- ESCM^2 [5]

这类方法通常通过共享参数、条件概率分解或者联合建模等方式，让稠密目标为稀疏目标提供额外监督信号，从而显著提升稀疏目标的学习效果。

6.3.5 行为关联建模

除了漏斗关系之外，不同互动行为之间本身也往往存在较强的相关性。例如在短视频和直播场景中，当用户对某位创作者产生较高兴趣时，点赞、评论、收藏以及关注等行为往往会共同出现。从概率角度来看，可以认为这些行为之间并非相互独立，而是存在潜在的因果关系或者兴趣传递关系：

$$Like \rightarrow Comment \rightarrow Follow \tag{6.7}$$

或者：

$$Like \rightarrow Favorite \rightarrow Follow \tag{6.8}$$

如果能够显式建模这些行为之间的依赖关系，往往能够提升多个目标的整体预测效果。基于这一思想，工业界提出了大量行为关系建模方法。例如美团提出的 AITM（Adaptive Information Transfer Multi-task framework）[6] 模型，便通过信息传递机制显式建模不同任务之间的依赖关系，从而解决传统多任务学习中任务之间相互干扰的问题。这一类挖掘多个目标之间相关性以及隐式关联的方法，在工业界也被称为“**多目标联合建模**”或者“**多目标感知建模**”即让多个稀疏目标在建模的时候能够互相感知到其他的目标信号，不仅在信号层面形成补充，也能够让模型挖掘出不同信号之间的隐式联系。

除此之外，近几年来出现的 PLE（Progressive Layered Extraction）[4]、MMoE（Multi-gate Mixture-of-Experts）[1]、SNR（Sub-Network Routing）[2] 等多任务学习框架，也都在一定程度上尝试挖掘不同目标之间的共享信息和差异化信息，从而提升稀疏目标的学习效果。

总体而言，稀疏互动目标建模是现代推荐系统精排模型中的核心研究方向之一。围绕样本不平衡、行为漏斗建模、目标关联建模以及多任务学习等问题，学术界和工业界已经提出了大量经典方法。这里只对相关思路进行了简要介绍，后续章节将进一步深入分析 ESMM、AITM、MMoE、PLE 等代表性模型的设计原理及其在工业界中的应用实践。

6.4 精排多目标融合

精排多目标融合阶段与粗排阶段的目标基本一致，其核心任务都是将多个模型输出的不同目标预估值（pxtr）进行统一融合，并依据融合分完成最终排序。在粗排阶段，由于模型结构相对简单，参与融合的因子通常以粗排主模型输出的若干个 pxtr 为主；而在精排阶段，由于候选集规模进一步缩小，系统能够容纳更多复杂模型参与打分，因此融合阶段需要处理的排序因子数量也会显著增加。

除了精排主模型输出的 CTR、EVTR、LTR、CVR 等基础预估值之外，工业界通常还会引入大量专门针对特定业务目标构建的 LTR 模型，例如：D2Q 时长预估模型、用户长期价值（LTV）模型、用户兴趣挖掘模型、基于人工经验 Reward 设计的 LTR 模型、Session 级强化学习模型、创作者价值模型、内容质量评估模型、新内容冷启动模型等。

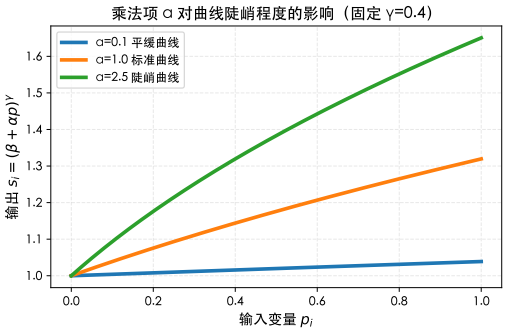
在大型工业界推荐系统中，参与融合的排序因子数量往往达到数十个甚至上百个。因此工业界通常会采用统一的多目标融合框架对这些因子进行管理与组合，例如加法公式、乘法公式以及基于序（Rank）的融合方式等。这一类融合与排序过程通常统称为 **Ensemble Sort (ES)**，即利用多个模型的排序能力进行集成学习，从而获得优于单模型排序的整体效果。

6.4.1 乘法公式及形状分析

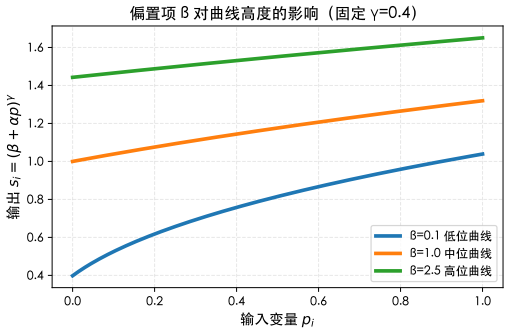
在工业界众多融合方法中，乘法公式（Multiplicative Formula）是一种应用非常广泛的融合框架。其基本形式如下：

$$s_i = (\beta_i + \alpha_i p_i)^{\gamma_i} \tag{6.9}$$

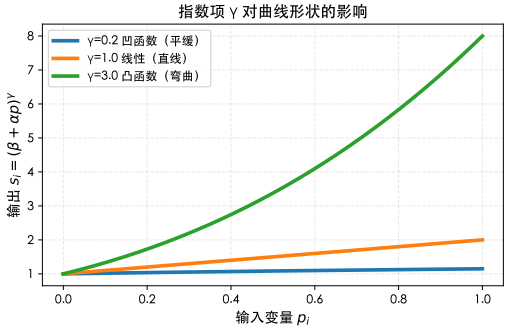
其中，p_i 表示第 i 个排序因子的原始预估值，α_i 为乘法系数（Scaling Factor），β_i 为偏置项（Bias Term），γ_i 为指数项（Exponent Term）。



(a) 乘法项 α 的影响



(b) 偏置项 β 的影响



(c) 指数项 γ 的影响

图 6.2: 乘法公式 $s_i = (\beta_i + \alpha_i p_i)^{\gamma_i}$ 中三个超参数的影响对比

图6.2展示了三个超参数对于最终得分曲线的影响。从工程实践角度来看，这三个参数实际上分别对应着排序系统中的三个重要控制维度：区分度控制（α_i）、基准权重控制（β_i）与流量分配形状控制（γ_i）。其中，乘法系数 α_i 主要负责调节原始 pxtr 的放大倍数。当 α_i 较大时，不同物品之间原本微小的 pxtr 差异会被进一步放大，从而提高排序结果的区分能力。此时系统更容易将流量集中分配给头部物品。反之，当 α_i 较小时，不同物品之间的得分差异被压缩，排序结果趋于平滑，系统整体探索性会有所增强。偏置项 β_i 则主要用于控制该因子的基础贡献水平。由于许多 pxtr 本身取值较小，例如点赞率、评论率、分享率等稀疏目标，如果直接参与乘法融合，很容易导致整体得分迅速衰减。因此，工业界通常会引入偏置项来稳定数值分布，同时调节不同目标在融合过程中的基础影响力。

相比之下，指数项 γ_i 往往是整个乘法公式中最重要的超参数。其本质决定了排序系统对于头部物品和尾部物品的偏好程度。当 $\gamma_i > 1$ 时，变换函数呈现凸函数（Convex Function）形态。此时高分物品会被进一步放大，而低分物品增长缓慢。因此系统会更倾向于选择在该目标上表现特别优秀的物品。从流量分发角度来看，这种配置会进一步强化头部效应（Matthew Effect）。当 $0 < \gamma_i < 1$ 时，函数呈现凹函数（Concave Function）形态。此时高分与低分物品之间的差距被压缩，系统会给予更多中腰部物品曝光机会，从而提升推荐结果的多样性和探索性。特殊地，当 $\gamma_i = 1$ 时，

$$(\beta_i + \alpha_i p_i)^{\gamma_i} = \beta_i + \alpha_i p_i \tag{6.10}$$

公式退化为线性变换，不再具有额外的非线性放大能力。总体而言：

- α_i 决定区分度；
- β_i 决定基础贡献；
- γ_i 决定流量分配形状。

因此，在工业界实践中，指数项 γ_i 通常也是最需要重点调优的参数。

6.4.2 序变换及形状分析

除了前述基于分数值（Value）的乘法融合公式之外，工业界还广泛使用基于排序位置（Rank）的融合方法。其核心思想是不直接利用 pxtr 的具体数值，而是仅利用不同候选物品在各个目标上的排序位置进行融合。

这种方法的出发点非常简单：不同模型输出的 pxtr 往往存在数值尺度不一致、分布漂移以及校准误差等问题。例如 CTR 的取值可能集中在 $[0.01, 0.2]$ 区间，而 CVR 的取值可能集中在 $[10^{-4}, 10^{-2}]$ 区间。如果直接利用数值进行融合，则需要频繁调整权重以适应不同因子的分布变化。而 Rank 本身只保留相对顺序信息，因此天然具有更强的鲁棒性。

为了对不同排序位置赋予不同的重要程度，工业界通常会设计各种非线性序变换函数。本章节选取以下五种具有代表性的变换方式进行分析：

$$\text{(1) 倒数变换：} \quad S_i = \frac{1}{\text{Rank}_i} \tag{6.11}$$

$$\text{(2) 平滑倒数变换：} \quad S_i = \frac{w_i}{\text{Rank}_i^{\lambda} + b} \tag{6.12}$$

$$\text{(3) 归一化幂次变换：} \quad S_i = w_i \left(1 - \left(\frac{\text{Rank}_i}{N}\right)^{\lambda}\right) \tag{6.13}$$

$$\text{(4) 指数双曲变换：} \quad S_i = w_i \left(e^{k\left(2\frac{\text{Rank}_i}{N} - 1\right)} - e^{-k\left(2\frac{\text{Rank}_i}{N} - 1\right)}\right) \tag{6.14}$$

$$\text{(5) 带偏置归一化变换：} \quad S_i = w_i \left(1 - \left(\frac{\text{Rank}_i}{N + b}\right)^{\lambda}\right) \tag{6.15}$$

其中，Rank_i 表示候选物品在某个排序因子中的位置，N 表示候选集长度，w_i 表示对应因子的权重，λ、b 和 k 为控制曲线形状的超参数。

上述五种典型序变换函数的曲线形状如图 6.3 所示。

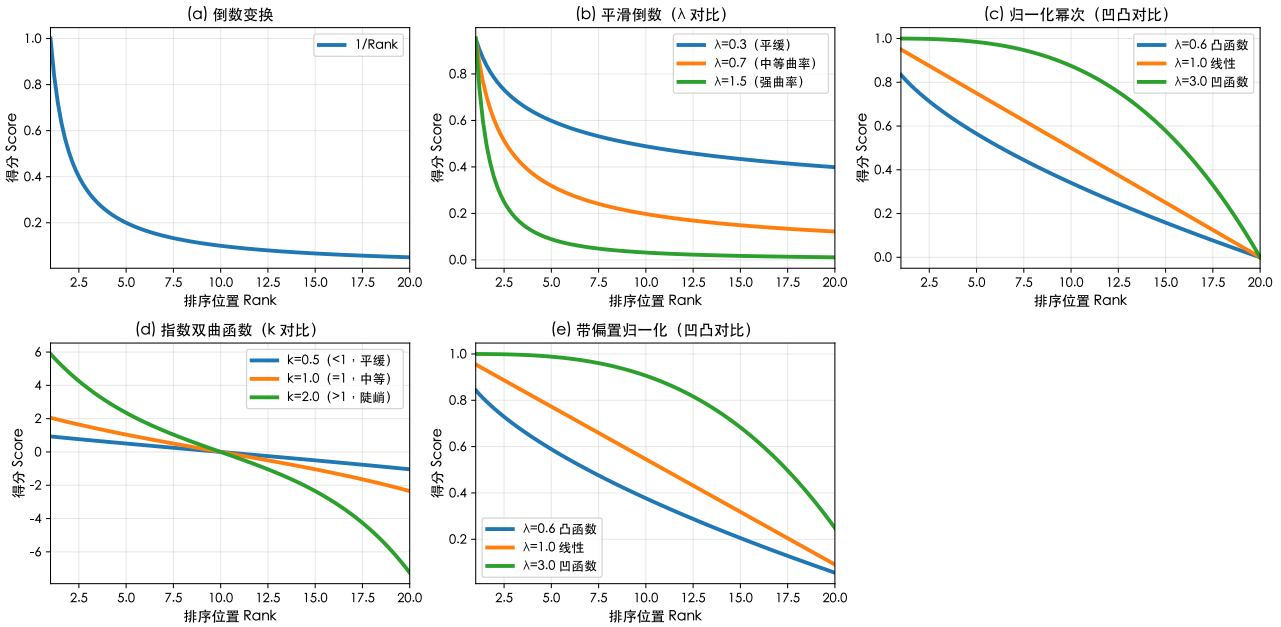


图 6.3: 不同排序位置变换函数的得分曲线对比。(a) 倒数变换；(b) 平滑倒数变换；(c) 归一化幂次变换；(d) 指数双曲变换；(e) 带偏置归一化变换。

从图 6.3(a) 可以看出，传统倒数变换是一种非常激进的头部强化策略。由于

$$\frac{1}{1}=1,\quad \frac{1}{2}=0.5,\quad \frac{1}{10}=0.1 \tag{6.16}$$

因此得分会随着 Rank 的增加快速衰减。这意味着排序靠前的少量物品能够获得远高于其他物品的融合得分。在实际业务中，这种变换往往会强烈偏向于某个目标上的 Top-K 候选物品。例如 CTR 排名第一和第二的物品之间差异巨大，而第 50 名和第 60 名之间几乎没有区别。因此倒数变换本质上是一种典型的 Head-Oriented 排序策略。

图 6.3(b) 展示了平滑倒数变换的效果。相比传统倒数变换，其引入了超参数 λ 和偏置项 b 用于调节曲线形状。当 $\lambda=0.3$ 时，曲线下降速度较慢，不同 Rank 之间的得分差异相对平缓；当 $\lambda=0.7$ 时，曲线开始体现明显的头部强化效果；而当 $\lambda=1.5$ 时，曲线在前几个位置急剧下降，Rank 大于 5 之后得分迅速趋近于零。这意味着较大的 λ 会显著提升 Top-K 物品的优势，使得某个排序因子的头部候选更容易在最终 ES 融合中胜出。因此，从工程经验来看，λ 本质上控制的是排序系统对于头部物品的偏好程度。

图 6.3(c) 和图 6.3(e) 分别展示了归一化幂次变换以及带偏置归一化变换。与倒数类函数不同，这两类函数能够更加灵活地控制整个排序区间内的得分分布。当 $\lambda<1$ 时，曲线整体呈现凸函数形态，此时头部区域下降速度较慢，而尾部区域下降速度较快。排序系统会更加关注 Top-K 候选物品之间的差异。当 $\lambda=1$ 时，曲线退化为线性形式。而当 $\lambda>1$ 时，曲线呈现凹函数特征。此时头部物品之间的差异被压缩，而中腰部物品之间的区分度得到增强。

因此这类函数最大的优势在于能够灵活控制排序资源究竟应该集中在头部区域还是分散到中腰部区域。例如在一些探索性推荐场景中，过度强化头部可能会导致内容同质化，此时往往会采用较大的 λ 来压平头部差异，从而增加中腰部内容的曝光机会。

图 6.3(d) 中的指数双曲变换则体现了另一种思路。其本质上对应双曲正弦函数：

$$S_i=2w_i\sinh\left(k\left(2\frac{\text{Rank}_i}{N}-1\right)\right) \tag{6.17}$$

与前面几种单调曲率函数不同，双曲函数在不同区间会表现出不同的敏感性。以 $k=2.0$ 为例，可以观察到：

- Rank 位于 Top-5 时，曲线下降非常迅速，强化头部候选之间的区分度；
- Rank 位于中腰部区间时，曲线变化趋于平缓，降低中部候选之间的差异；
- Rank 位于尾部区域时，曲线再次变陡，从而重新拉开尾部候选之间的距离。

这种特性使得双曲函数能够实现局部区域的差异化建模。例如有些业务希望重点区分 Top-K 和尾部内容，而对中间区域不那么敏感，此时双曲函数往往能够取得更好的效果。

总体来看，不同序变换函数本质上对应着不同的流量分配哲学。倒数变换强调头部优先；平滑倒数变换强调可控的头部强化；幂次变换强调头中尾资源的重新分配；双曲变换则能够实现不同排序区域的差异化调节。因此在工业界实践中，并不存在绝对最优的序变换函数。最终采用何种曲线形状，通常需要结合具体业务目标，通过大量线上 A/B 实验不断验证和迭代。

最后，在讲完了乘法公式变换和序变换以及它们对应的排序公式形状分析之后，我们可以发现，不同的 Rank 变换函数本质上都在解决同一个问题：

笔记 如何分配不同排序位置之间的区分能力？

有些函数更关注头部物品之间的排序精度；有些函数更关注中腰部物品之间的竞争关系；还有一些函数会刻意强化尾部物品之间的区分能力，从而为探索流量提供更多空间。因此这些变换函数本质上并不存在绝对优劣，其效果最终取决于具体业务场景、候选集分布以及线上流量分发目标。在工业界实践中，策略算法工程师通常会结合 AB 实验，通过超参数搜索、自动化调参以及强化学习等方法寻找最优的 Rank 变换形状，而不是依赖固定的经验公式。

6.4.3 排序公式超参数寻优

在粗排多目标融合章节中，我们介绍了加法公式、乘法公式以及基于序变换的融合公式。这些公式虽然能够较好地融合多个 pxtr 因子，但往往包含大量超参数，例如加法公式中的权重参数、乘法公式中的偏置项与指数项，以及序变换中的曲率参数等。传统工业界通常采用人工经验调参的方式确定这些超参数。然而随着排序因子数量不断增加，一个实际的排序公式往往包含几十甚至上百个待优化参数。此时依赖人工经验进行调参不仅效率较低，而且难以找到全局最优解。因此近年来越来越多的推荐系统开始引入**自动化搜参（Auto Tuning）**技术，其中最经典的方法之一便是 **Cross-Entropy Method（CEM）**算法。

6.4.3.0.1 CEM 算法基本思想 假设排序公式包含 N 个待优化参数：

$$\boldsymbol{\theta} = (\theta_1, \theta_2, \cdots, \theta_N) \tag{6.18}$$

我们的目标是寻找最优参数：

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) \tag{6.19}$$

其中 $R(\boldsymbol{\theta})$ 表示线上 AB 实验所反馈的业务收益指标，例如：

- 人均观看时长；
- 用户留存率；
- GMV；
- 广告收益；
- 长期价值（LTV）等。

由于线上指标通常无法直接求导：

$$\frac{\partial R}{\partial \theta_i} \tag{6.20}$$

因此传统梯度下降方法无法直接应用。

CEM 的核心思想是：**不直接搜索最优参数，而是搜索最优参数所在的概率分布。**假设参数服从一个多维高斯分布：

$$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma) \tag{6.21}$$

，其中：

$$\boldsymbol{\mu} = (\mu_1, \mu_2, \cdots, \mu_N) \tag{6.22}$$

表示参数均值，Σ 表示协方差矩阵。CEM 算法并不直接优化参数本身，而是不断更新：

$$(\boldsymbol{\mu}, \Sigma) \tag{6.23}$$

使采样分布逐渐向高收益区域收缩。

6.4.3.0.2 采样阶段 在第 t 轮迭代时，从当前分布中采样：

$$\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_K \tag{6.24}$$

满足：

$$\boldsymbol{\theta}_i \sim \mathcal{N}(\boldsymbol{\mu}^{(t)}, \Sigma^{(t)}) \tag{6.25}$$

每一个采样点都对应一套排序公式参数。例如：

$$\boldsymbol{\theta}_1 = (w_1, w_2, \gamma_1, \gamma_2, \cdots) \tag{6.26}$$

代表一个具体的多目标融合配置。

6.4.3.0.3 线上评估阶段 对于每组参数，都分配一部分线上流量进行实验。经过若干小时后获得对应收益：

$$R_1, R_2, \cdots, R_K \tag{6.27}$$

，然后按照收益排序：

$$R_{(1)} \geq R_{(2)} \geq \cdots \geq R_{(K)} \tag{6.28}$$

并保留前 ρ 比例的优秀样本：

$$\mathcal{E} = \{\boldsymbol{\theta}^*(1), \cdots, \boldsymbol{\theta}^*(M)\} \tag{6.29}$$

，其中：

$$M = \rho K \tag{6.30}$$

。通常来说，ρ 参数满足：$\rho \in [0.05, 0.2]$。这一部分样本被称为 Elite Samples（精英样本）。

6.4.3.0.4 分布更新阶段 在分布更新阶段，我们利用精英样本重新估计高斯分布参数。需要对均值和协方差分别进行更新。其中，均值更新如下：

$$\boldsymbol{\mu}^{(t+1)} = \frac{1}{M} \sum_{i=1}^{M} \boldsymbol{\theta}_{(i)} \tag{6.31}$$

，协方差更新如下：

$$\Sigma^{(t+1)} = \frac{1}{M} \sum_{i=1}^{M} (\boldsymbol{\theta}_{(i)} - \boldsymbol{\mu}^{(t+1)})(\boldsymbol{\theta}_{(i)} - \boldsymbol{\mu}^{(t+1)})^T \tag{6.32}$$

这样新的采样分布会逐渐向高收益区域移动。随着迭代不断进行，会有：

$$\Sigma^{(t)} \to 0 \tag{6.33}$$

，即采样空间不断收缩，最终收敛到一个局部最优解附近。

需要注意的是，CEM 并不是简单的随机搜索。其理论基础来源于最小化两个概率分布之间的 KL Divergence：

$$D_{KL}(g^*\|f_\theta) \tag{6.34}$$

，其中：g^* 表示理想的最优解分布，f_θ 表示当前采样分布。最小化 KL Divergence 等价于最小化交叉熵（Cross Entropy），因此该方法被称为 Cross-Entropy Method。

6.4.3.0.5 工业界在线搜参流程 在推荐系统实际落地过程中，CEM 通常用于排序公式自动调优。例如，对于一个乘法融合公式：

$$Score = \prod_{i=1}^{N}(\beta_i + \alpha_i p_i)^{\gamma_i} \tag{6.35}$$

，其中：

$$\boldsymbol{\theta} = (\alpha_1, \cdots, \alpha_N, \beta_1, \cdots, \beta_N, \gamma_1, \cdots, \gamma_N) \tag{6.36}$$

全部作为待搜索参数。

系统首先随机生成数百组参数配置，然后将线上流量切分给不同实验桶。经过数小时甚至数天的数据积累后，根据观看时长、留存率、GMV 等指标选出收益最好的若干组参数，再利用这些参数重新拟合采样分布并继续搜索。经过多轮迭代之后，参数空间会逐渐向收益更高的区域收缩，从而自动获得优于人工调参的排序公式。

从本质上看，CEM 并不是在优化排序公式本身，而是在不断学习：

$$P(\boldsymbol{\theta} \mid R\text{较高}) \tag{6.37}$$

即**“高收益排序公式最有可能出现在哪些参数区域”**。因此，CEM 可以被看作一种基于概率分布迭代更新的黑盒优化方法，在推荐系统、多目标融合、强化学习策略搜索以及自动机器学习（AutoML）等领域均有广泛应用。

6.4.3.0.6 工业界 CEM 应用时存在的问题 虽然从理论上来看，CEM 能够通过不断迭代更新采样分布逐步逼近最优参数区域，但在工业界推荐系统的实际应用过程中，我们发现其效果往往没有理论分析中那么理想。究其原因，问题并不完全出在优化算法本身，而更多来自推荐系统环境本身所具有的复杂性与不确定性。

首先，推荐系统的在线反馈数据天然具有较大的噪声（Noise）。在理想情况下，我们希望每组超参数对应一个准确且稳定的收益指标：$R(\theta)$。然而在线系统中的收益往往受到大量随机因素影响。例如用户的实时兴趣变化、节假日效应、热点事件、内容供给变化、流量波动甚至埋点异常等因素都会影响最终观测到的指标。因此实际观测到的收益通常更接近于：

$$\hat{R}(\theta) = R(\theta) + \epsilon \tag{6.38}$$

，其中：ϵ 表示各种来源的随机噪声。这意味着某一组参数在线上表现更好，并不一定是因为参数本身更优，也有可能仅仅是采样到了更有利的流量或者更活跃的用户群体。因此 CEM 每一轮筛选出的 Elite Samples 并不一定是真正意义上的最优参数区域，也有可能只是在局部最优做随机扰动。

其次，推荐系统中的用户反馈往往存在明显的**延迟反馈（Delay Feedback）**问题。例如观看时长、点赞、评论等行为通常能够在分钟级甚至秒级获得反馈，但很多更重要的业务指标却需要较长时间才能观测到。例如：

- 次日留存（Day-1 Retention）；
- 七日留存（Day-7 Retention）；
- 用户生命周期价值（LTV）；
- 长期 GMV；
- 用户长期满意度。

这些指标往往需要数天甚至数周的数据积累才能较为准确地评估。

因此在实际应用中会出现一个典型矛盾：

- 如果使用小时级反馈指标进行寻优，则反馈速度快，迭代效率高；
- 如果使用天级甚至周级反馈指标进行寻优，则目标更准确，但优化周期极长。

例如对于一轮 CEM 搜索，如果每次评估需要等待一天：

$$T_{eval} = 1\ Day \tag{6.39}$$

假设需要执行 20 轮迭代：

$$T_{total} = \#20 \times T_{eval} = 20\ Days \tag{6.40}$$

那么整个搜索过程可能需要数周时间才能完成。而在这期间，用户兴趣分布、内容生态以及流量结构可能已经发生明显变化，此时得到的最优参数甚至可能已经不再适用于当前系统。

进一步地，随着推荐系统复杂度不断提升，多目标融合公式中的参数规模也在迅速增长。假设一个乘法融合公式包含 $N = 50$ 个排序因子。对于每个因子分别需要优化：

$$\alpha_i, \beta_i, \gamma_i \tag{6.41}$$

这三个参数。则整体参数空间规模达到：$3N = 150$ 维。当参数维度不断增加时，搜索空间会呈指数级膨胀：

$$\mathcal{O}(c^N) \tag{6.42}$$

即所谓的**维度灾难（Curse of Dimensionality）**。此时即使采用 CEM，也往往需要大量采样才能覆盖有效搜索区域，因此搜索效率会迅速下降。很多情况下，算法甚至还没有收敛，业务环境已经发生变化。

正因为如此，近年来工业界开始探索更加自动化和智能化的排序公式优化方案。例如强化学习（Reinforcement Learning）方法尝试直接学习排序策略，而非搜索固定参数；Bayesian Optimization 则利用代理模型预测高收益区域；而随着大语言模型（LLM）的兴起，一些公司也开始尝试利用 LLM Agent 自动分析实验结果、生成新的排序策略，并进行超参数调整和实验规划。从形式上看，LLM Agent 的优化流程与传统 CEM 存在明显区别：

- CEM 基于概率分布进行随机搜索；
- Bayesian Optimization 基于代理模型进行搜索；
- 强化学习通过策略梯度学习最优策略；
- LLM Agent 则试图利用推理能力自动设计下一轮实验。

这些方法看起来越来越智能，也越来越自动化。但从工业界实践经验来看，它们最终仍然需要面对同样的问题：

- 数据噪声是否足够小；
- 反馈信号是否足够及时；
- 优化目标是否真正反映业务价值；
- 在线实验是否具有统计显著性；
- 用户长期价值是否被正确衡量。

事实上，在推荐系统自动寻优过程中，一个经常被忽视但又最关键的问题是：

笔记 *究竟什么指标才应该作为自动寻优的目标函数?*

例如短视频推荐场景中：

- 小时级观看时长增长是否代表长期收益增长?
- 次日留存增长是否一定代表长期满意度提升?
- 互动率提升是否会损害用户体验?
- GMV 提升是否会降低内容生态健康度?

这些问题往往没有标准答案。

因此无论是 CEM、强化学习还是 LLM Agent，本质上都只是优化器（Optimizer）。真正决定系统最终效果

的，仍然是目标函数设计、数据质量以及业务理解能力。很多时候，如何定义一个合理的优化目标，远比选择哪一种优化算法更加重要。这也是工业界推荐系统自动寻优领域至今仍然持续研究和探索的重要方向。

6.5 精排流量调控

精排流量调控（Traffic Control）是精排模块中的最后一个核心环节。在完成精排主模型、LTR 模型以及多目标融合之后，系统已经能够得到每个候选物品对应的融合排序分数。然而，在真实的工业场景中，仅仅按照融合分从高到低直接选取 Top-K 物品往往并不能满足业务需求。因此，在最终结果输出之前，通常还需要经过一层流量调控模块，对候选集合进行进一步约束与优化。

6.5.1 级联式多目标融合与流量调控

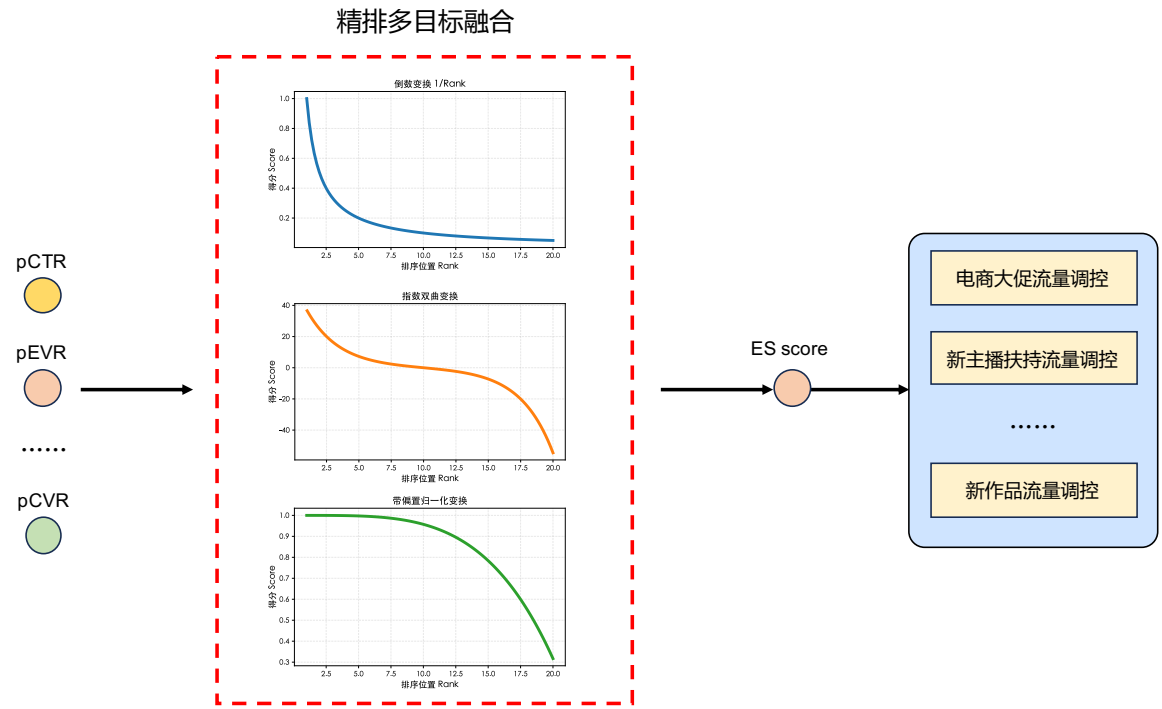


图 6.4: 精排级联式多目标融合与流量调控模块框架。

从系统架构视角来看，工业界精排链路的流量调控暂无统一落地规范，业内主流实现思路是将**多目标融合**与**流量调控**解耦为两个串行独立阶段，整体架构如图 6.4 所示。

系统首先利用精排模型与 LTR 模型输出的各类 PXTR 指标完成多目标融合。在融合之前，通常会先对各类 PXTR 进行非线性变换，再通过加法公式、乘法公式等方式得到统一排序分数。随后按照融合分完成初始排序，并将排序结果送入流量调控模块。

流量调控阶段主要负责处理模型难以直接学习的业务规则，例如新内容扶持、新作者成长保护、电商活动流量倾斜、商业化保量等。其常见实现方式包括动态 Boost、局部 Rank 调整、Quota 控制以及保量策略等。在这一架构下，排序逻辑与业务逻辑相互独立，具有实现简单、维护成本低以及业务扩展灵活等优点，可以精细化地实现定向的业务流量倾斜，因此被大量工业系统采用。

下面给出一个典型的级联式流量调控伪代码示例：

代码 6.1: 级联式多目标融合与流量调控伪代码。

```
class RankItem:
    def __init__(self, pctr, pwatch_time, plike, is_new_video=False, author_fans=0,
```

```
                in_campaign=False):
        self.pctr = pctr
        self.pwatch_time = pwatch_time
        self.plike = plike

        self.is_new_video = is_new_video
        self.author_fans = author_fans
        self.in_campaign = in_campaign

        self.score = 0.0

class TrafficController:
    def __init__(self, ctr_weight, wt_weight, like_weight):
        self.ctr_weight = ctr_weight
        self.wt_weight = wt_weight
        self.like_weight = like_weight

    def multi_objective_fusion(self, item):
        return (
            self.ctr_weight * item.pctr + self.wt_weight * item.pwatch_time +
            self.like_weight * item.plike
        )

    def traffic_control(self, item):
        score = item.score
        # 新内容扶持
        if item.is_new_video:
            score *= 1.10
        # 新作者扶持
        if item.author_fans < 1000:
            score *= 1.05
        # 电商活动扶持
        if item.in_campaign:
            score *= 1.15
        return score

    def rank(self, candidates, topk=100):
        # Step1 多目标融合
        for item in candidates:
            item.score = self.multi_objective_fusion(item)

        # Step2 初始排序
        rank_list = sorted(
            candidates,
            key=lambda x: x.score,
            reverse=True
        )

        # Step3 流量调控
```

```
    for item in rank_list:
        item.score = self.traffic_control(item)

    # Step4 再排序
    rank_list = sorted(rank_list, key=lambda x: x.score, reverse=True)

    # Step5 输出TopK
    return rank_list[:topk]
```

可以看到，在这种级联式多目标融合与流量调控架构下，排序模型首先给出统一排序结果，而流量调控作为后置模块对排序结果进行修正。这种方式实现简单，排序逻辑与业务逻辑解耦程度较高，也是工业界最常见的实现方式。

6.5.2 多通道式流量调控

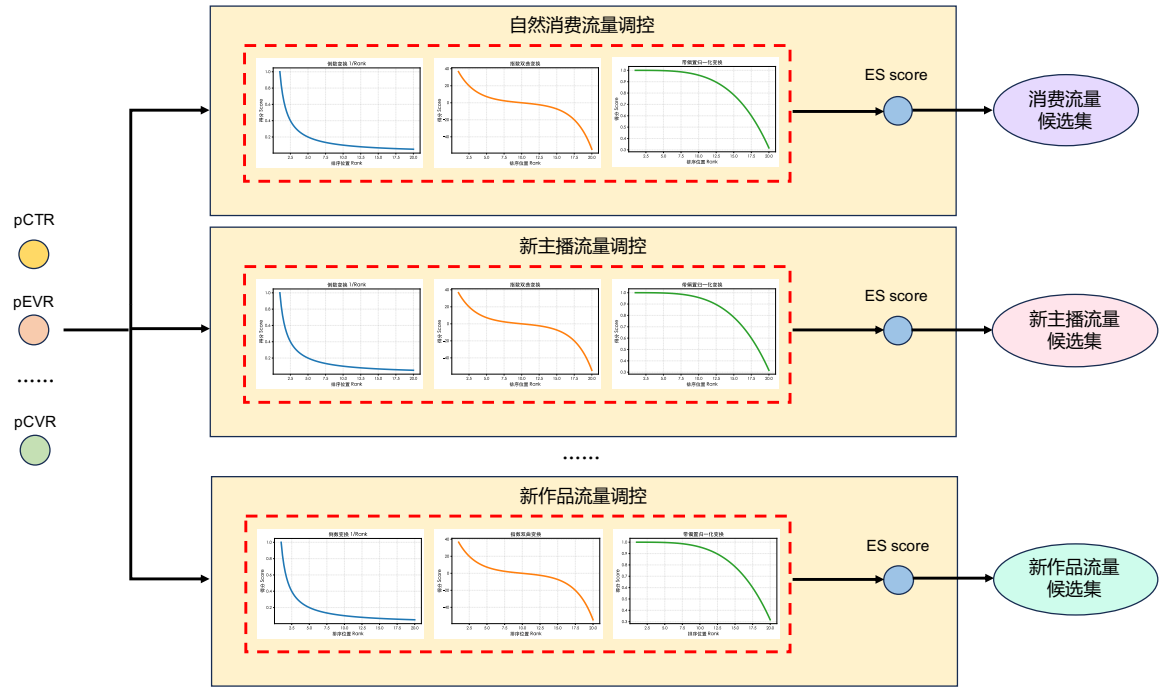


图 6.5: 精排多通道式流量调控模块框架。

另一种常见方案则是借鉴粗排阶段的多通道（Multi-Channel）或多 Quota 架构，如下图6.5所示。在这种架构下，系统首先根据不同业务目标构建多个独立候选池，例如新内容池、电商内容池、创作者扶持池以及普通内容池等。每个候选池内部拥有独立的排序逻辑、多目标融合策略以及流量优化目标。随后，各通道分别完成内部排序，再按照预先设定的 Quota 进行结果合并，共同组成最终的 Top-K 推荐结果。与级联式架构相比，多通道架构将流量调控能力前置到了候选池构建阶段，多目标融合与流量分配往往被统一设计和联合优化，因此对于复杂业务场景具有更强的表达能力。

下面给出多通道式流量调控的伪代码示例：

代码 6.2: 多通道式流量调控

```
from typing import List

class RankItem:
    def __init__(self, pctr, pwatch_time, plike, is_new_video=False, author_fans=0,
```

```
                in_campaign=False):
        self.pctr = pctr
        self.pwatch_time = pwatch_time
        self.plike = plike

        self.is_new_video = is_new_video
        self.author_fans = author_fans
        self.in_campaign = in_campaign

        self.score = 0.0


class ChannelRanker:
    def __init__(self, ctr_weight, watch_weight, like_weight):
        self.ctr_weight = ctr_weight
        self.watch_weight = watch_weight
        self.like_weight = like_weight

    def score(self, item):
        return (
            self.ctr_weight * item.pctr + self.watch_weight * item.pwatch_time +
            self.like_weight * item.plike
        )

    def rank(self, items: List[RankItem]):
        for item in items:
            item.score = self.score(item)
        return sorted(items, key=lambda x: x.score, reverse=True)

class MultiChannelTrafficControl:
    def __init__(self):
        self.new_video_ranker = ChannelRanker(ctr_weight=0.3, watch_weight=0.4, like_weight=0.3)
        self.commerce_ranker = ChannelRanker(ctr_weight=0.2, watch_weight=0.2, like_weight=0.6)
        self.creator_ranker = ChannelRanker(ctr_weight=0.4, watch_weight=0.4, like_weight=0.2)
        self.normal_ranker = ChannelRanker(ctr_weight=0.5, watch_weight=0.3, like_weight=0.2)

    def split_channel(self, candidates):
        new_pool = []
        commerce_pool = []
        creator_pool = []
        normal_pool = []

        for item in candidates:
            if item.is_new_video:
                new_pool.append(item)
            elif item.in_campaign:
                commerce_pool.append(item)
            elif item.author_fans < 1000:
                creator_pool.append(item)
```

```
            else:
                normal_pool.append(item)
        return new_pool, commerce_pool, creator_pool, normal_pool

    def rank(self, candidates):
        new_pool, commerce_pool, creator_pool, normal_pool = self.split_channel(candidates)

        # 通道内部独立排序
        new_pool = self.new_video_ranker.rank(new_pool)
        commerce_pool = self.commerce_ranker.rank(commerce_pool)
        creator_pool = self.creator_ranker.rank(creator_pool)
        normal_pool = self.normal_ranker.rank(normal_pool)

        # Quota控制
        result = []
        result.extend(new_pool[:10])
        result.extend(commerce_pool[:20])
        result.extend(creator_pool[:15])
        result.extend(normal_pool[:55])
        return result
```

多通道架构能够更加自然地表达复杂业务目标，因此在直播推荐、电商推荐以及内容生态治理等场景中被广泛采用。但与此同时，其系统复杂度、参数维护成本以及流量分配策略设计难度也明显高于级联式方案。

6.5.3 流量调控的业务价值

从业务目标视角而言，精排阶段的流量调控核心价值在于调和**模型最优**与**业务最优**之间的天然矛盾。模型输出的排序分值仅拟合历史数据的统计分布规律，大量落地层面的业务约束难以被模型直接建模；若仅依靠 pxtr 指标经非线性融合后的分值筛选 Top-K 候选，该分数仅刻画用户消费行为偏好，无法承载繁杂的业务管控逻辑。

工业推荐系统常见的业务约束可归纳为：新内容冷启动扶持，低粉丝创作者流量保护，新晋主播成长资源倾斜，电商大促货品保量投放，商业化广告曝光保底，优质稀缺内容探索导流，长尾内容基础曝光兜底，各垂类内容分发均衡管控，用户推荐结果多样性约束等。上述生态与运营类目标难以嵌入模型损失函数做联合优化，因此需要在精排打分完成后，依托流量调控模块做显式后置约束。以短视频推荐为例，纯模型驱动的排序容易使搞笑、娱乐等高点击内容垄断曝光，知识、生活、体育等垂类内容分发受限，虽短期核心指标利好，但会持续破坏平台内容生态；直播场景同理，完全依照预估收益排序会不断虹吸流量向头部主播集中，新晋与中腰部主播缺少成长空间，最终造成平台生态僵化。

由此可见，流量调控承担着平台生态治理（Ecosystem Governance）的关键职能。这也体现了工业推荐系统的复杂性：用户即时转化、点击率预估这类用户行为规律可交由模型拟合优化，而刚性业务规则、运营配额约束与平台生态平衡等诉求，则需要通过策略调控的方式落地实现。

6.5.4 流量调控的优化视角

另一方面，从优化问题的角度来看，精排流量调控实际上可以被看作一个带约束条件的排序优化问题：

$$\max \sum_{i=1}^{N} Score_i \tag{6.43}$$

同时满足：

$$\sum_i x_i^{new} \geq Q_{new} \tag{6.44}$$

$$\sum_i x_i^{creator} \geq Q_{creator} \tag{6.45}$$

$$\sum_i x_i^{commerce} \geq Q_{commerce} \tag{6.46}$$

，其中 $x_i \in \{0,1\}$ 表示候选物品是否被最终选中，Q 表示对应业务目标要求的最小流量配额。因此，精排流量调控实际上是在最大化排序收益与满足业务约束之间寻找平衡点。

近年来随着强化学习（Reinforcement Learning）、组合优化（Combinatorial Optimization）以及大模型 Agent 技术的发展，一些先进推荐系统已经开始尝试将流量调控问题建模为动态决策问题。系统不再依赖人工配置固定 Quota，而是根据实时流量状态、内容供给情况以及用户反馈动态调整流量分配策略。例如：

$$Q_t = f(S_t^{user}, S_t^{Content}, S_t^{Platform}) \tag{6.47}$$

其中，Quota Q_t 会随着实时环境变化而动态更新。

不过从目前工业界实践来看，大多数线上系统仍然以规则策略、Quota 控制以及轻量级优化框架为主，而强化学习和大模型 Agent 类方案更多应用于部分复杂业务场景或者离线辅助决策系统之中。

6.5.5 精排保量与强插策略

除了前面介绍的基于 Quota 的流量调控机制之外，工业界推荐系统中还广泛存在另一类更加直接的流量控制方式，即保量（Guarantee Delivery）机制。与 Quota 更多关注不同内容池之间的流量分配比例不同，保量机制关注的是某一类特定候选集合是否能够获得最低限度的曝光机会，从而满足平台的业务目标和运营需求。

在实际业务中，推荐系统并不总是单纯追求短期点击率、观看时长等指标的最大化。对于一些处于萌芽期的新业务、新内容形态或者平台重点扶持的方向，如果完全依赖模型排序，其流量往往会被成熟业务和头部内容挤压，难以获得成长空间。因此，平台通常会为这些业务设置专门的保量策略，确保其能够获得稳定的曝光机会。例如新上线的内容频道、新的视频互动功能、新的电商业务形态等，都可能在早期通过保量机制获得冷启动流量支持。

此外，保量机制也广泛应用于创作者生态建设中。例如平台运营团队筛选出的优质创作者、优质主播、重点签约作者或者官方活动参与创作者等，往往会进入特定的白名单池。对于这些候选集合，系统会根据业务目标设置最低曝光量、最低曝光人数或者最低流量占比等约束条件，从而保证优质内容能够被用户发现，而不会因为短期数据不足而被排序模型完全压制。

从实现方式来看，保量机制通常可以表示为一种带约束的排序优化问题。在最终结果生成过程中，除了优化整体排序收益之外，还需要满足：

$$Exposure_{target} \geq Q_{target} \tag{6.48}$$

其中，$Exposure_{target}$ 表示目标候选集合实际获得的曝光量，Q_{target} 表示业务要求的最低保量目标。

在工程实践中，保量可以通过多种方式实现。例如在排序完成后进行强插（Force Insert），直接将符合条件的候选物品插入最终结果；也可以在流量调控阶段预留固定坑位；或者将保量约束直接融入多目标融合与流量分配框架之中，通过动态调整排序分数实现更加平滑的流量控制。保量机制本质上是在模型最优与业务最优之间寻找平衡，一方面推荐系统需要充分利用模型能力提升用户体验和核心指标；另一方面也需要兼顾业务发展、内容生态建设以及平台长期价值。因此，保量机制往往是精排流量调控模块中不可或缺的重要组成部分，也是策略算法与产品运营深度结合的典型体现。

总体而言，精排流量调控并不仅仅是简单的保量或者 Quota 控制模块，而是连接算法目标与平台业务目标的重要桥梁。一方面需要尽可能保留模型排序带来的收益，另一方面又需要兼顾内容生态、创作者成长、商业化

目标以及长期用户价值等复杂约束。正因如此，流量调控往往也是推荐系统中策略算法工程师参与最深、业务理解要求最高的模块之一。

6.6 本章小结

本章我们详细介绍了级联式推荐系统架构中的精排模块及其核心设计思想。在上一章中，我们已经介绍了粗排模块如何从海量候选集中构建高质量候选集合。而精排模块则进一步承担起用户兴趣建模与最终排序决策的职责，其核心目标是在粗排提供的候选集合基础上，更加准确地评估用户对每个候选物品的真实兴趣与潜在价值，从而输出高质量的排序结果。

首先，我们介绍了精排模块的整体架构，包括精排主模型与 Learning to Rank（LTR）模型、多目标融合以及流量调控等核心组成部分。其中，精排主模型负责预测用户对于候选物品的各种行为反馈，包括点击、观看时长、点赞、评论、转发、关注以及转化等行为目标。在实际工业场景中，由于评论、关注、分享等行为往往具有极强的稀疏性，因此如何有效提升稀疏目标的建模能力成为精排模型优化的重要方向。本章围绕这一问题，介绍了样本加权、稠密目标辅助学习以及行为依赖关系建模等多种常见方案，并以 ESSM、AITM 等典型方法为例进行了说明。

随后，我们重点讨论了精排阶段的多目标融合问题。由于推荐系统往往同时面临用户体验、内容生态以及商业化收益等多个优化目标，因此需要将多个 pxtr 以及业务目标统一映射到同一排序空间中。本章从排序公式设计的角度，对工业界广泛使用的乘法融合公式进行了深入分析，并重点讨论了偏置项 β、缩放项 α 以及指数项 γ 对排序结果和流量分布的影响。同时，我们还介绍了基于序关系的多目标融合方法，包括倒数变换、平滑倒数变换、归一化幂函数变换、指数双曲变换以及带偏置项的归一化变换等常见形式，并从函数形状的角度分析了不同公式对于流量分配策略的影响。

在排序公式设计完成之后，如何高效地寻找最优超参数同样是工业界的重要问题。针对这一问题，本章介绍了排序公式超参数自动寻优的基本思路，并重点讲解了 Cross-Entropy Method（CEM）这一常用的黑盒优化算法，为后续复杂排序策略的自动化调参奠定基础。

最后，我们介绍了精排阶段的流量调控模块。与粗排阶段相比，精排阶段距离最终结果更近，因此流量调控策略对于线上指标和业务目标的影响也更加直接。本章分别介绍了级联式多目标融合与流量调控框架，以及多通道式流量调控框架两种常见设计方案。同时，我们还简要讨论了保量策略、强插策略等工程实践中广泛使用的流量干预手段，以及它们在用户体验、内容生态与商业化目标之间所起到的平衡作用。

总体而言，精排模块是整个推荐系统中最核心的排序决策模块。粗排阶段主要关注候选集合的构建，而精排阶段则更加关注单个物品价值的精确评估与排序。工业界大量先进的用户兴趣建模、多任务学习、多目标优化以及流量调控技术，也大多围绕精排阶段展开迭代与演进。

下一章我们将继续介绍级联式推荐系统中的重排模块。与粗排和精排主要关注单个物品排序不同，重排模块通常从列表（List）或序列（Slate）的角度进行整体优化，更加关注推荐结果之间的相互关系、多样性以及长期收益等问题。不过，重排模块并非所有推荐系统都必须具备。例如在直播推荐场景中，如果一次请求最多只返回一个直播间，则通常不需要额外的重排阶段；而在短视频、信息流、电商等一次请求需要返回多个结果的场景中，重排模块则往往发挥着十分重要的作用。

6.7 参考文献

[1] Jiaqi Ma et al. “Modeling task relationships in multi-task learning with multi-gate mixture-of-experts”. In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, pp. 1930–1939.

[2] Jiaqi Ma et al. “Snr: Sub-network routing for flexible parameter sharing in multi-task learning”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 216–223.

[3] Xiao Ma et al. “Entire space multi-task model: An effective approach for estimating post-click conversion rate”. In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 2018, pp. 1137–1140.

[4] Hongyan Tang et al. “Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations”. In: *Proceedings of the 14th ACM conference on recommender systems*. 2020, pp. 269–278.

[5] Hao Wang et al. “ESCM2: Entire space counterfactual multi-task model for post-click conversion rate estimation”. In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2022, pp. 363–372.

[6] Dongbo Xi et al. “Modeling the sequential dependence among audience multi-step conversions with multi-task learning in targeted display advertising”. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2021, pp. 3745–3755.