第 20 章 用户兴趣与序列建模

从本书第11章《粗排模型》和第12章《精排模型》的介绍中可以看到，用户兴趣建模是工业级推荐系统粗排与精排模型的重要组成部分。用户的兴趣不仅体现在历史交互行为中，还会随着时间推移、行为反馈和内容消费的变化而动态演化。因此，如何从用户的历史行为中提取有效的兴趣表示，并根据当前候选物品进行针对性的兴趣建模，是提升推荐模型效果的重要研究方向。

本章围绕用户兴趣建模与行为序列建模的主要技术路线展开，分别介绍基于对比学习、基于 Attention、基于 Memory、基于召回检索、基于哈希以及基于时序图的用户行为序列建模方法。这些方法从不同角度解决历史行为的表示学习、目标感知兴趣提取、长序列计算效率以及复杂行为关系建模等问题，体现了工业推荐系统在用户兴趣建模方面的多种技术思路。

除了改进序列建模方法本身，如何进一步扩大用户行为序列的建模规模（Scaling Up）也是近年来的重要研究方向。尤其是在面对超长用户行为序列时，如何在控制计算与存储开销的同时，充分利用更丰富的历史行为信息，成为构建大规模推荐模型的关键问题。近年来，LONGER [2]、Climber [26] 等方法围绕大规模推荐模型的高效扩展展开探索，为用户行为序列的 Scaling Up 提供了新的思路。相关内容将在后续第36章《推荐大模型》中进一步介绍。

20.1 基于对比学习的用户行为序列建模

对比学习（Contrastive Learning）是一种通过构造正负样本关系学习数据表示的方法，其核心思想如下：

定义 20.1

对比学习的目标是拉近语义相近样本在表示空间中的距离，同时拉远不相关样本之间的距离。与传统监督学习主要依赖单个样本的标签优化预测结果不同，对比学习通过样本之间的相对关系提供额外的监督信号，使模型能够学习更加具有区分性的特征表示。

在推荐系统中，用户的历史交互行为包含丰富的兴趣信息，但不同类型的行为往往具有不同的语义。例如，用户对物品的点击、长播观看、短播观看以及主动负反馈等行为，可能分别反映兴趣探索、正向偏好与负向反馈。只依赖 CTR 等单一预测目标，模型未必能够充分区分这些行为背后的兴趣差异。对比学习可以通过构造不同类型的正负样本，为用户行为序列的 Embedding 提供额外的优化目标，使模型更好地学习不同兴趣和反馈之间的关系。

根据正负样本的构造方式，对比学习可以分为基于不同样本实例的对比、基于数据增强视图的对比，以及结合用户行为反馈进行的对比学习等类型。本节首先介绍 InfoNCE、SimCLR、MoCo 和 SimSiam 等具有代表性的对比学习方法，分析不同方法的基本原理及其在表示学习中的应用。随后，本节结合工业推荐系统中的实际需求，介绍如何利用正负反馈行为序列构建对比学习目标，以及如何通过行为序列增广进一步增强用户兴趣表示。

20.1.1 InfoNCE 对比学习

InfoNCE（Noise-Contrastive Estimation）是对比学习中最经典的损失函数之一，最早由 2018 年的论文《Representation Learning with Contrastive Predictive Coding》[18] 在 Contrastive Predictive Coding（CPC）框架中提出。该方法的核心思想如下：

定义 20.2

InfoNCE 对比学习通过构造正样本和负样本，将表示学习转换为样本区分任务，使模型在拉近相关样本表示的同时，拉远不相关样本表示，从而学习具有较强区分能力的特征 Embedding。

传统的监督学习主要依赖样本标签优化预测结果，而在许多场景中，大量数据并没有明确的人工标注。即使存在用户行为日志，也不一定能够直接获得足够精细的监督信号。相比之下，数据本身往往蕴含着丰富的关联关系，例如同一段语音中相邻时间片段之间具有较强的相关性，同一张图像中不同区域之间存在共同的语义信息，用户近期连续消费的物品也可能反映相近的兴趣偏好。InfoNCE 希望利用这些关联关系构造训练信号，从而学习能够保留关键语义信息的表示。

20.1.1.0.1 基于互信息的表示学习 InfoNCE 的理论基础与互信息（Mutual Information，MI）密切相关。互信息用于衡量两个随机变量之间共享信息的多少，可以表示为：

$$I(x;c) = \sum_{x,c} p(x,c) \log \frac{p(x|c)}{p(x)} \tag{20.1}$$

其中，$p(x,c)$ 表示两个随机变量的联合概率分布，$p(x|c)$ 表示给定 c 时 x 的条件概率，而 $p(x)$ 表示 x 的边缘概率。互信息越大，说明已知变量 c 能够提供越多关于变量 x 的信息。

对于表示学习的任务，通常希望通过神经网络将原始输入映射到低维 Embedding 空间，并尽可能保留不同输入之间具有共同意义的信息。例如，在用户行为序列建模中，用户过去的交互行为能够反映其兴趣偏好，而未来发生的行为通常也与这些兴趣存在一定关联。如果模型能够学习过去行为表示与未来行为表示之间的共享信息，就有机会提取用户相对稳定的潜在兴趣，同时减少局部噪声和偶然行为的影响。

基于这一思想，CPC 提出了一种通过预测未来潜在表示来学习特征的方法。与传统生成模型直接预测原始输入不同，CPC 希望最大化两个潜在表示之间的互信息，使模型重点学习对未来预测有价值的特征。由于潜在表示是由原始输入经过编码得到的，根据数据处理不等式，编码后的表示之间的互信息不会超过原始信号之间的互信息。因此，通过最大化潜在表示的互信息，可以促使模型保留原始数据中具有共享关联的信息。

20.1.1.0.2 Contrastive Predictive Coding CPC 是 InfoNCE 最初提出时所采用的表示学习框架，其主要包含两个部分：非线性编码器（Encoder）和自回归模型（Autoregressive Model）。首先，CPC 利用编码器将原始输入序列中的每个观测映射为潜在表示：

$$z_t = g_{\text{enc}}(x_t) \tag{20.2}$$

其中，x_t 表示时刻 t 的原始输入，g_{enc} 表示非线性编码网络，z_t 表示对应的潜在 Embedding。随后，自回归模型对截至当前时刻的历史潜在表示进行聚合，得到上下文表示：

$$c_t = g_{\text{ar}}(z_{\leq t}) \tag{20.3}$$

其中，$z_{\leq t}$ 表示从起始时刻开始到 t 时刻的所有潜在表示，g_{ar} 表示自回归网络，c_t 则包含当前时刻之前的上下文信息。

在得到上下文表示之后，CPC 希望利用 c_t 预测未来时刻 $t+k$ 对应的潜在表示 z_{t+k}。这里并不是直接重建原始输入 x_{t+k}，而是判断未来潜在表示与当前上下文是否匹配。为此，模型采用密度比（Density Ratio）作为预测目标：

$$f_k(x_{t+k}, c_t) \propto \frac{p(x_{t+k}|c_t)}{p(x_{t+k})} \tag{20.4}$$

其中，$p(x_{t+k}|c_t)$ 表示给定当前上下文时未来观测出现的条件概率，$p(x_{t+k})$ 表示未来观测的边缘概率。密度比越大，说明当前未来观测越符合已有上下文所蕴含的信息，而不是一个随机出现的观测。

在具体实现中，CPC 使用一个简单的双线性函数计算上下文与未来潜在表示之间的匹配分数：

$$f_k(x_{t+k}, c_t) = \exp(z_{t+k}^T W_k c_t) \tag{20.5}$$

其中，W_k 为预测未来第 k 个时间步对应的可训练矩阵，不同预测步长可以使用独立的参数。通过这种方式，模型不需要直接生成高维的未来观测，而只需要学习未来潜在表示与当前上下文之间的匹配关系，从而降低表示学习的建模难度。

图20.1展示了 CPC 的整体结构。原始输入首先经过非线性编码器得到潜在表示，再通过自回归网络聚合历史信息形成上下文向量。模型以该上下文为依据，对未来多个时间步的潜在表示进行预测，并通过正负样本之

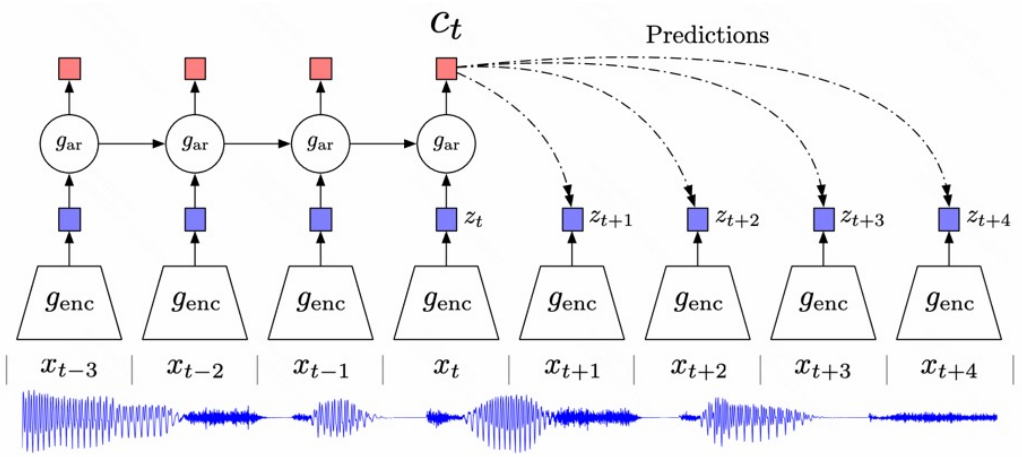


图 20.1: CPC 的整体结构示意图。

间的对比学习优化编码器和自回归网络。论文以音频信号为例进行展示，但这一结构同样可以应用于图像、文本以及强化学习等不同类型的数据。

20.1.1.0.3 InfoNCE 损失函数 在 CPC 中，如何衡量未来潜在表示与当前上下文之间的匹配程度，是训练过程中的关键问题。直接计算条件概率 $p(x_{t+k}|c_t)$ 通常需要对高维数据建立复杂的概率分布模型，计算成本较高。因此，CPC 采用基于噪声对比估计（Noise-Contrastive Estimation，NCE）的方式，将概率建模转化为正负样本之间的判别任务，并由此提出 InfoNCE 损失函数。

具体来说，对于给定的上下文表示 c_t，构造包含 N 个样本的候选集合 $\mathcal{X}$，集合中一个样本来自真实的条件分布 $p(x_{t+k}|c_t)$ 作为正样本；而其余 $N-1$ 个样本从边缘分布 $p(x_{t+k})$ 中随机采样作为负样本：

$$\mathcal{X} = \{x_{t+k}, x_1^-, x_2^-, \ldots, x_{N-1}^-\} \tag{20.6}$$

其中 x_{t+k} 表示与当前上下文对应的真实未来观测，x_j^- 表示从边缘分布中采样得到的负样本。模型需要从这 N 个候选样本中识别出唯一的正样本。

根据前面定义的匹配函数 f_k，模型可以将正样本的匹配分数与所有候选样本的匹配分数进行归一化，得到正样本被正确识别的概率。由此，InfoNCE 损失可以表示为：

$$\mathcal{L}_N = -\mathbb{E}_{\mathcal{X}}\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in \mathcal{X}} f_k(x_j, c_t)}\right] \tag{20.7}$$

从计算形式来看，InfoNCE 本质上是一个多分类交叉熵损失。对于每个上下文，模型需要在一个正样本和多个负样本之间进行选择，最大化真实正样本对应的归一化匹配概率。由于 f_k 通常采用指数形式的匹配分数，因此也可以将上述目标理解为在正负样本之间执行 Softmax 分类。

为了理解 InfoNCE 为什么能够学习有效的样本表示，可以进一步分析该损失函数的最优解。首先，假设给定上下文 c_t，构造一个包含 N 个候选样本的集合 $\mathcal{X} = \{x_1, \ldots, x_N\}$，其中一个样本从条件分布 $p(x|c_t)$ 中采样，作为正样本；其余 $N-1$ 个样本则从边缘分布 $p(x)$ 中独立采样，作为负样本。为了避免模型直接通过样本位置识别正负样本，将正样本随机放置在候选集合中的任意位置。定义随机变量 d 表示正样本所在的位置，即 $d = i$ 表示 x_i 是正样本。在这种采样方式下，每个位置成为正样本的先验概率相同，即：

$$p(d = i|c_t) = \frac{1}{N} \tag{20.8}$$

接下来，考虑给定正样本位置 $d = i$ 时，整个候选集合 $\mathcal{X}$ 出现的概率。由于 x_i 来自条件分布 $p(x|c_t)$，其余 $N-1$ 个样本均来自边缘分布 $p(x)$，并且各个样本独立采样，因此整个候选集合出现的概率等于各个样本采样概率的乘积：

$$p(\mathcal{X}|d = i, c_t) = p(x_i|c_t)\prod_{j \neq i} p(x_j) \tag{20.9}$$

另一方面，在仅给定上下文 c_t 而不知道正样本具体位置的情况下，候选集合 $\mathcal{X}$ 可能由 N 种不同的采样过程产生：第一个位置是正样本，第二个位置是正样本，依此类推，直到第 N 个位置是正样本。这些情况彼此互斥，并且覆盖了正样本可能出现的全部位置。因此，根据全概率公式，候选集合出现的边缘概率可以表示为对所有可能正样本位置对应概率的加权求和：

$$p(\mathcal{X}|c_t) = \sum_{k=1}^{N} p(\mathcal{X}, d=k|c_t) \tag{20.10}$$

进一步利用联合概率的乘法公式，可以将上式展开为：

$$p(\mathcal{X}|c_t) = \sum_{k=1}^{N} p(\mathcal{X}|d=k, c_t)p(d=k|c_t) \tag{20.11}$$

由于正样本在 N 个位置中均匀随机放置，因此 $p(d=k|c_t) = 1/N$。再将前面得到的候选集合条件概率代入，可以得到：

$$p(\mathcal{X}|c_t) = \frac{1}{N}\sum_{k=1}^{N} p(x_k|c_t)\prod_{j\neq k} p(x_j) \tag{20.12}$$

例如，当候选集合中包含三个样本时，上述全概率公式可以具体展开为：

$$p(\mathcal{X}|c_t) = \frac{1}{3}\left[p(x_1|c_t)p(x_2)p(x_3) + p(x_1)p(x_2|c_t)p(x_3) + p(x_1)p(x_2)p(x_3|c_t)\right] \tag{20.13}$$

其中，三项分别对应第一个、第二个和第三个样本作为正样本时，生成整个候选集合的概率。由此可以看到，全概率公式的作用就是将正样本位于不同位置的所有可能情况进行汇总，得到在未知正样本位置时观察到候选集合的总概率。

在有了候选集合的边缘概率之后，就可以利用贝叶斯公式计算正样本位于第 i 个位置的后验概率：

$$p(d=i|\mathcal{X}, c_t) = \frac{p(\mathcal{X}|d=i, c_t)p(d=i|c_t)}{p(\mathcal{X}|c_t)} \tag{20.14}$$

将前面推导得到的条件概率和边缘概率代入，并约去分子与分母中的公共因子 $1/N$，可以得到：

$$p(d=i|\mathcal{X}, c_t) = \frac{p(x_i|c_t)\prod_{j\neq i} p(x_j)}{\sum_{k=1}^{N} p(x_k|c_t)\prod_{j\neq k} p(x_j)} \tag{20.15}$$

为了进一步简化表达式，将分子和分母同时除以所有候选样本边缘概率的乘积 $\prod_{j=1}^{N} p(x_j)$。对于分子中的第 i 项，有：

$$\frac{p(x_i|c_t)\prod_{j\neq i} p(x_j)}{\prod_{j=1}^{N} p(x_j)} = \frac{p(x_i|c_t)}{p(x_i)} \tag{20.16}$$

同理，对于分母中的每一项，都可以消去除对应正样本以外的所有边缘概率，最终得到：

$$p(d=i|\mathcal{X}, c_t) = \frac{\frac{p(x_i|c_t)}{p(x_i)}}{\sum_{k=1}^{N}\frac{p(x_k|c_t)}{p(x_k)}} \tag{20.17}$$

可以看到，正样本的后验概率由条件概率 $p(x_i|c_t)$ 与边缘概率 $p(x_i)$ 的比值决定。该比值反映了样本在给定上下文后出现的概率相对于随机出现概率的变化程度，能够衡量样本与上下文之间的关联性。对于在当前上下文中更容易出现的样本，其概率密度比通常更大，也就更有可能被判断为正样本。

另一方面，InfoNCE 通过匹配函数 $f(x, c_t)$ 构造模型预测的正样本概率：

$$p_f(d=i|\mathcal{X}, c_t) = \frac{f(x_i, c_t)}{\sum_{j=1}^{N} f(x_j, c_t)} \tag{20.18}$$

由于 InfoNCE 本质上是针对正样本位置的分类交叉熵，当模型具有足够的表示能力且达到最优解时，其预测的后验概率应当与真实后验概率一致。比较两者的表达式，可以得到最优匹配函数满足：

$$f^*(x, c_t) \propto \frac{p(x|c_t)}{p(x)} \tag{20.19}$$

这说明 InfoNCE 并不是简单地要求正样本与上下文在表示空间中更加接近，而是通过区分真实关联样本和随机负样本，使模型学习到能够反映二者统计关联程度的概率密度比。

20.1.1.0.4 互信息下界 InfoNCE 除了能够用于构造正负样本的分类目标之外，还具有一个重要的理论性质：其损失函数可以用于构造两个随机变量之间互信息的下界。对于包含一个正样本和 $N-1$ 个负样本的候选集合，InfoNCE 满足：

$$I(x_{t+k}; c_t) \geq \log N - \mathcal{L}_N \tag{20.20}$$

其中，N 表示候选样本总数，包含一个正样本和 $N-1$ 个负样本。由此可见，最小化 InfoNCE 损失等价于最大化互信息的一个下界，使编码后的未来表示与上下文表示尽可能保留更多共享信息。

从这个角度来看，InfoNCE 将原本难以直接计算和优化的互信息目标，转化为一个可以通过正负样本采样实现的分类损失。通常情况下，增加负样本数量能够提高互信息下界的紧致程度，但同时也会带来更多的计算开销。此外，负样本的采样分布还会影响模型学习到的区分关系。如果负样本过于容易区分，模型可能无法充分学习细粒度的语义差异。反之，如果负样本与正样本在实际语义上高度相似，则可能引起模型的误判。

在工业级推荐系统中，InfoNCE 损失给学习用户行为之间的关联关系以及构建高质量 Embedding 提供了一个通用的优化目标。例如，我们可以将用户历史行为中的相邻物品、同一兴趣下的不同交互或者同一行为序列经过不同数据增强后得到的表示构造成正样本，而将其他物品或不同兴趣下的行为表示作为负样本。通过对比学习，模型能够在表示空间中保留用户行为之间具有区分性的关联信息，为后续的用户兴趣提取和行为序列建模提供更好的 Embedding 基础。

总体来看，InfoNCE 的核心思想是将表示学习转换为正负样本之间的判别任务，通过 Softmax 归一化构造可优化的对比损失，并利用负样本估计真实样本与上下文之间的密度比。其互信息下界性质进一步为这种优化方式提供了理论依据。由于该损失函数不依赖特定的数据模态或网络结构，只需要能够**构造具有明确关联关系的正负样本**，就可以应用于图像、文本、语音及推荐系统等多种场景，也因此成为后续 SimCLR、MoCo 等对比学习方法的重要基础。

20.1.2 SimCLR 对比学习

SimCLR 源自于 Google 研究团队发表在 ICML 2020 会议的论文《A Simple Framework for Contrastive Learning of Visual Representations》[7]。该工作提出了一种简单而有效的对比学习框架，通过对同一个样本进行不同的数据增强，构造具有相同语义的正样本对，再利用其他样本的增强视图作为负样本，学习具有区分性的特征表示。

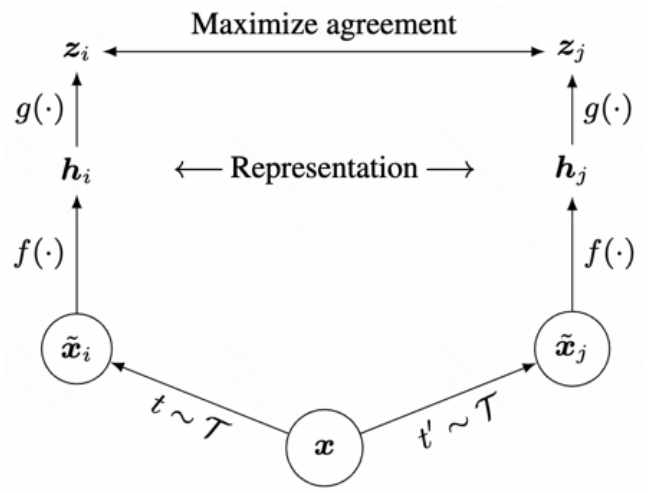


图 20.2: SimCLR 的对比学习结构。

图20.2展示了 SimCLR 的整体结构。在训练阶段，模型通过数据增强构造正样本对，并利用基础编码网络和投影头共同优化对比损失。当训练完成后，投影头可以直接移除，仅保留基础编码网络 $f(\cdot)$，将其输出作为下游任务的特征表示。SimCLR 的整体结构主要包含四个部分：数据增强模块、基础编码网络（Base Encoder）、投影头（Projection Head）以及对比学习损失。首先，对于一个原始样本 x，从相同的数据增强分布 $\mathcal{T}$ 中独立采样两

个增强操作 t 和 t'，可以得到两个不同的视图：

$$\widetilde{x}_i = t(x), \quad \widetilde{x}_j = t'(x), \quad t, t' \sim \mathcal{T} \tag{20.21}$$

两个视图来自同一个原始样本，因此构成正样本对。如果是针对图片的对比学习任务，SimCLR 则采用随机裁剪并缩放、随机颜色扰动以及高斯模糊等操作，使同一图像产生不同的视觉表示。

随后，SimCLR 利用共享参数的基础编码网络 $f(\cdot)$ 提取两个视图的特征表示，并通过一个额外的 MLP 投影头 $g(\cdot)$ 将其映射到对比学习空间，即：

$$\mathbf{h}_i = f(\widetilde{x}_i), \quad \mathbf{z}_i = g(\mathbf{h}_i) = \mathbf{W}^{(2)}\sigma(\mathbf{W}^{(1)}\mathbf{h}_i) \tag{20.22}$$

其中 $\mathbf{h}_i$ 表示基础编码网络输出的特征表示，$\mathbf{z}_i$ 表示经过投影头得到的对比学习表示，$\sigma(\cdot)$ 表示 ReLU 激活函数。对于一个包含 N 个原始样本的 Mini-batch，经过两次数据增强后可以得到 $2N$ 个视图。其中当前样本与增广样本构成 2 组正样本对，而其余 $2(N-1)$ 个视图均作为负样本对。这里，SimCLR 采用归一化温度缩放交叉熵（Normalized Temperature-scaled Cross Entropy，NT-Xent）作为对比学习损失：

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{I}(k \neq i) \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)} \tag{20.23}$$

其中，$\text{sim}(\cdot,\cdot)$ 表示余弦相似度，τ 为温度系数，$\mathbb{I}(\cdot)$ 为指示函数，用于排除当前锚点自身。最终，模型对一个 Mini-batch 中的所有正样本对进行双向计算，并取平均得到整体损失：

$$\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell_{2k-1,2k} + \ell_{2k,2k-1}] \tag{20.24}$$

通过上述方式，SimCLR 在同一个 Mini-batch 内构造正负样本，而不需要额外的负样本采样器或 Memory Bank。一个 Mini-batch 中的其他增强视图都可以作为负样本，因此增大 Batch Size 能够增加可用负样本的数量。总体来看，SimCLR 通过简单的数据增强和 Batch 内负样本构造，实现了无需人工标签的特征表示学习。该方法的重要贡献在于将数据增强、特征编码和投影头进行解耦，并通过大批量训练充分利用负样本信息，为后续对比学习方法提供了一个简洁而有效的基础框架。

20.1.3 MoCo 对比学习

MoCo 源自于 Facebook AI 团队发表在 CVPR 2020 会议的论文《Momentum Contrast for Unsupervised Visual Representation Learning》[13]。该工作针对对比学习中负样本数量受 Mini-batch 规模限制的问题，提出了动量对比学习（Momentum Contrast，MoCo）方法，通过队列（Queue）和动量更新的编码器构建一个规模较大的负样本字典，在保证特征表示一致性的同时，降低对大批量训练的依赖。

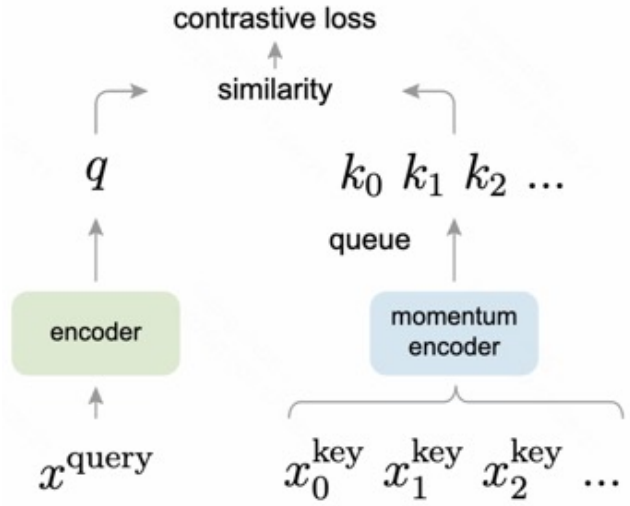


图 20.3: MoCo 的动量对比学习结构。

图20.3展示了 MoCo 的整体结构。通过将负样本字典设计为动态队列，并使用动量编码器生成具有较好一致性的 Key，模型能够在较小 Batch Size 下利用大量负样本进行对比学习。MoCo 将对比学习理解为一个字典查找（Dictionary Look-up）任务。假设当前输入经过编码得到查询向量（Query）$\mathbf{q}$，同时存在一个由多个编码向量组成的字典 $\{\mathbf{k}_0, \mathbf{k}_1, \ldots, \mathbf{k}_K\}$，其中 $\mathbf{k}_0$ 为与 Query 对应的正样本，其余 $\mathbf{k}_1, \ldots, \mathbf{k}_K$ 为负样本。模型希望 Query 与正样本 Key 具有较高的相似度，同时与其他 Key 保持较低的相似度。因此，MoCo 对应的 InfoNCE 损失为：

$$\mathcal{L}_q = -\log \frac{\exp(\mathbf{q}^T \mathbf{k}_0/\tau)}{\sum_{i=0}^{K} \exp(\mathbf{q}^T \mathbf{k}_i/\tau)} \tag{20.25}$$

其中，τ 为温度系数。该目标本质上是一个包含一个正样本和 K 个负样本的多分类任务。

在传统对比学习方法中，负样本通常来自当前 Mini-batch，因此负样本字典的规模受到 Batch Size 和显存容量的限制。为了解决这一问题，MoCo 引入队列机制，将最近若干个 Mini-batch 产生的 Key Embedding 存储在一个固定长度的队列中。每次训练时，将当前 Mini-batch 生成的新 Key 加入队列，同时移除队列中最早加入的 Key。这样，模型就可以利用多个历史 Mini-batch 中的样本构建负样本集合，而不需要一次性增大 Batch Size。

然而，直接使用队列还存在一个问题，即队列中的 Key 可能由不同训练时刻的编码器生成。如果编码器参数更新过快，不同时刻产生的 Embedding 就可能存在较大差异，导致负样本字典中的表示缺乏一致性。因此，MoCo 进一步设计了动量更新机制，分别使用 Query Encoder $f_q(\cdot)$ 和 Key Encoder $f_k(\cdot)$ 生成两路 Embedding。其中，Query Encoder 通过反向传播正常更新，而 Key Encoder 不直接参与梯度优化，而是根据 Query Encoder 的参数进行动量式更新：

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q \tag{20.26}$$

其中，θ_q 和 θ_k 分别表示两个编码器的参数，$m \in [0, 1)$ 表示动量系数。通过这种动量式更新，Key Encoder 的参数变化会更加平缓，从而使得不同时刻生成的 Key Embedding 保持较好的一致性。

以图像对比学习任务为例，在具体训练过程中，模型首先对同一图像执行两次随机数据增强，分别生成 Query 和正样本 Key。当前 Mini-batch 中的 Key 经过 Key Encoder 编码后加入队列，而 Query 则由 Query Encoder 生成。随后，模型利用当前 Query、正样本 Key 以及队列中的负样本计算 InfoNCE 损失，并更新 Query Encoder。在完成参数更新后，按照动量机制更新 Key Encoder，同时将当前 Mini-batch 的 Key 入队，并将最早的一批 Key 出队。

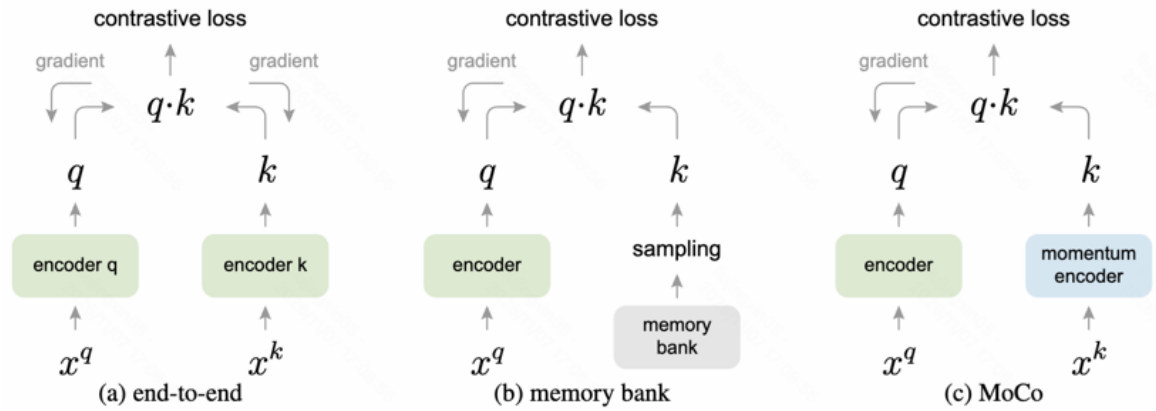


图 20.4: 三种对比学习负样本维护机制对比。

从负样本维护方式来看，MoCo 与传统的端到端对比学习以及 Memory Bank 方法也存在明显区别。传统端到端方法直接使用当前 Mini-batch 构建负样本集合，Query 和 Key 由相同训练阶段的编码器生成，表示一致性较好，但负样本数量受到 Batch Size 限制。Memory Bank 则将整个数据集的 Embedding 存储起来，每次随机抽取部分历史表示作为负样本，虽然可以获得较大的负样本集合，但不同样本的 Embedding 可能来自相距较远的训练时刻，表示一致性较差。MoCo 则结合了两者的特点，即：

笔记 Moco 使用队列维护大量负样本，同时通过动量编码器减缓 Key 表示的变化。

图20.4对比了这三种负样本维护机制。MoCo 通过队列和动量编码器，在负样本字典规模与表示一致性之间取得了较好的平衡。总的来说，MoCo 的主要贡献是将负样本字典与当前 Mini-batch 解耦，并通过动量更新机

制保证不同训练时刻产生的 Key Embedding 具有较好的连续性。与 SimCLR 依赖大批量训练获取负样本不同，MoCo 可以在相对较小的 Batch Size 下利用较大的负样本集合，从而降低训练时对显存和批量规模的要求。

20.1.4 SimSiam 对比学习

SimSiam 源自于 Facebook AI 团队发表在 CVPR 2021 会议的论文《Exploring Simple Siamese Representation Learning》[8]。该工作进一步研究了对比学习中负样本的作用，提出了一种不依赖负样本和动量编码器的自监督表示学习方法。与 SimCLR 和 MoCo 通过区分正负样本学习特征不同，SimSiam 只利用同一个样本经过不同数据增强得到的两个视图，通过预测网络和停止梯度机制（Stop-Gradient）学习一致的特征表示。

图20.5展示了 SimSiam 的整体结构。两个增强视图经过共享编码器得到特征表示，其中一路通过预测头生成预测结果，另一路通过 Stop-Gradient 作为学习目标。模型通过最大化两路表示的余弦相似度完成训练，整个过程不需要显式构造负样本，也不需要额外维护动量编码器。

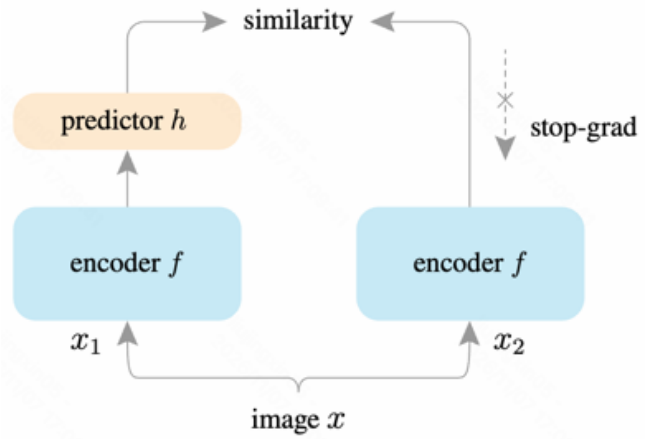


图 20.5: SimSiam 的孪生网络结构。

具体来说，对于一个原始样本 x，首先使用两种随机数据增强得到两个视图 x_1 和 x_2。随后，两个视图分别经过共享参数的编码器 $f(\cdot)$，得到对应的特征表示：

$$\mathbf{z}_1 = f(x_1), \quad \mathbf{z}_2 = f(x_2) \tag{20.27}$$

其中，编码器 $f(\cdot)$ 由基础网络（Backbone）和投影 MLP 组成。在此基础上，SimSiam 额外引入一个预测头（Prediction Head）$h(\cdot)$，对其中一路特征进行进一步变换，得到预测表示：

$$\mathbf{p}_1 = h(\mathbf{z}_1), \quad \mathbf{p}_2 = h(\mathbf{z}_2) \tag{20.28}$$

模型希望让一路预测表示与另一路编码表示保持较高的一致性。因此，采用负余弦相似度构造损失函数：

$$D(\mathbf{p}_1, \mathbf{z}_2) = -\frac{\mathbf{p}_1^T \mathbf{z}_2}{\|\mathbf{p}_1\|_2 \|\mathbf{z}_2\|_2} \tag{20.29}$$

为了避免模型只学习单向的表示匹配关系，SimSiam 进一步采用对称损失，同时计算两个方向的匹配结果：

$$\mathcal{L} = \frac{1}{2} D(\mathbf{p}_1, \mathbf{z}_2) + \frac{1}{2} D(\mathbf{p}_2, \mathbf{z}_1) \tag{20.30}$$

当只使用上述一致性损失时，模型可能出现表示坍塌（Representation Collapse），即不同输入最终都被映射为近似相同的 Embedding，使得两个分支之间的相似度虽然较高，却失去了有效的特征区分能力。为了解决这一问题，SimSiam 引入了 Stop-Gradient 操作，在计算损失时将目标分支的表示视为常量，不再向该分支反向传播梯度：

$$\mathcal{L} = \frac{1}{2} D(\mathbf{p}_1, \text{stopgrad}(\mathbf{z}_2)) + \frac{1}{2} D(\mathbf{p}_2, \text{stopgrad}(\mathbf{z}_1)) \tag{20.31}$$

这里的 Stop-Gradient 操作只会阻断当前损失项向目标分支的梯度传播，而不会完全停止该分支的网络训练。例如，在第一项损失中，$\mathbf{z}_2$ 不接收该项产生的梯度，但在第二项损失中，$\mathbf{z}_1$ 仍然可以通过预测表示 $\mathbf{p}_2$ 接收梯度。

因此，两个分支仍然能够通过对称损失来进行共同更新，只是在每个损失项中形成预测分支与目标分支之间不对称的梯度关系。

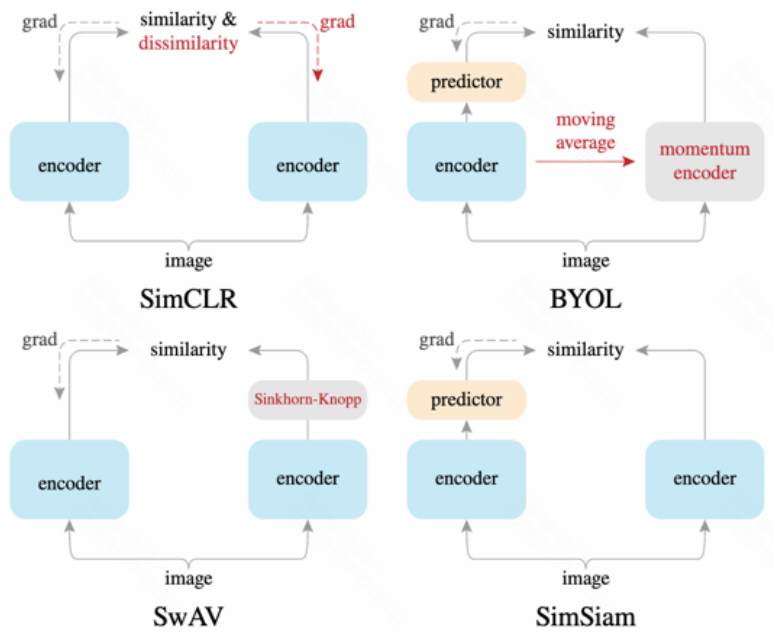


图 20.6: 不同孪生网络表示学习方法的结构对比。

从网络结构上看，SimSiam 与 SimCLR、MoCo 以及 BYOL、SwAV 等孪生网络表示学习方法存在一定联系。这些方法通常包含两个输入分支，通过不同的数据视图构造样本间的对应关系，但在预测头、动量编码器、负样本以及梯度传播方式等方面存在差异。SimSiam 的特点是保留了共享编码器和预测头，同时移除了负样本和动量编码器，仅通过 Stop-Gradient 实现两路表示之间的非对称优化。图20.6对比了几种典型孪生网络结构中的主要组件及其梯度传播关系，可以看到 SimSiam 通过去除部分复杂组件，构建了更加简洁的自监督表示学习框架。

总体来看，SimSiam 表明对比学习并不一定需要显式的负样本集合。通过共享编码器、预测头和 Stop-Gradient 的组合，也能够在不同增强视图之间学习具有区分性的表示。与 SimCLR 依赖批次内负样本以及 MoCo 利用动量队列扩大负样本集合不同，SimSiam 进一步简化了训练过程，为不依赖负样本的自监督表示学习提供了一种有效思路。

20.1.5 基于正负反馈行为序列的对比学习建模

以短视频推荐系统为例，用户的实时行为反馈是刻画兴趣变化的重要信号。尤其是短播行为，通常反映用户对当前内容缺乏足够兴趣，能够为模型识别用户当前不感兴趣的内容提供直接依据。例如，用户刚刚短时间浏览某一类型的视频后，如果精排模型仍然持续推荐相似内容，就容易重复触发用户的隐式负反馈，影响后续观看体验。

在工业推荐系统中，已有的实时反馈模块通常会利用用户近期短播的视频与当前候选视频来训练一个双塔模型，通过向量相似度计算来筛选出与短播行为相似的候选视频，并在排序阶段对其进行 deboost。然而，这类“双塔模型 + 负反馈策略打压”的方法往往依赖独立的 Embedding 模型，其优化目标与最终的排序目标可能并不完全一致。与此同时，传统双塔模型主要围绕点击、观看时长等正向预估目标进行训练，未必能够充分区分用户短播与非短播行为之间的差异。

因此，我们可以直接在精排模型中构建正负反馈行为序列，分别提取用户近期的正向兴趣和负向兴趣，并利用对比学习进一步约束候选物品与两类行为表示之间的关系。这样既能够保留原有精排模型的预测能力，也能够通过辅助监督增强模型对实时负反馈的识别能力。

在特征构造阶段，精排模型分别抽取截至当前请求时刻的近期短播序列 S^- 和非短播序列 S^+。其中，短播序列主要包含用户近期短时间观看的视频，非短播序列则包含有效播放或者长时间观看的视频。每条行为记录

可以包含视频 ID、播放时长、视频时长、视频标签及行为发生时间戳等信息。短播的判定需要同时结合播放时长和视频时长，具体阈值会根据业务场景来确定。正反馈和负反馈的两类序列按照行为发生时间排序，并通过截断和掩码处理变长输入。此外，精排模型还可以将反馈行为距离当前请求的时间间隔、播放时长占视频总时长的比例等信息作为辅助特征，使模型能够同时感知反馈的新近程度和观看深度。

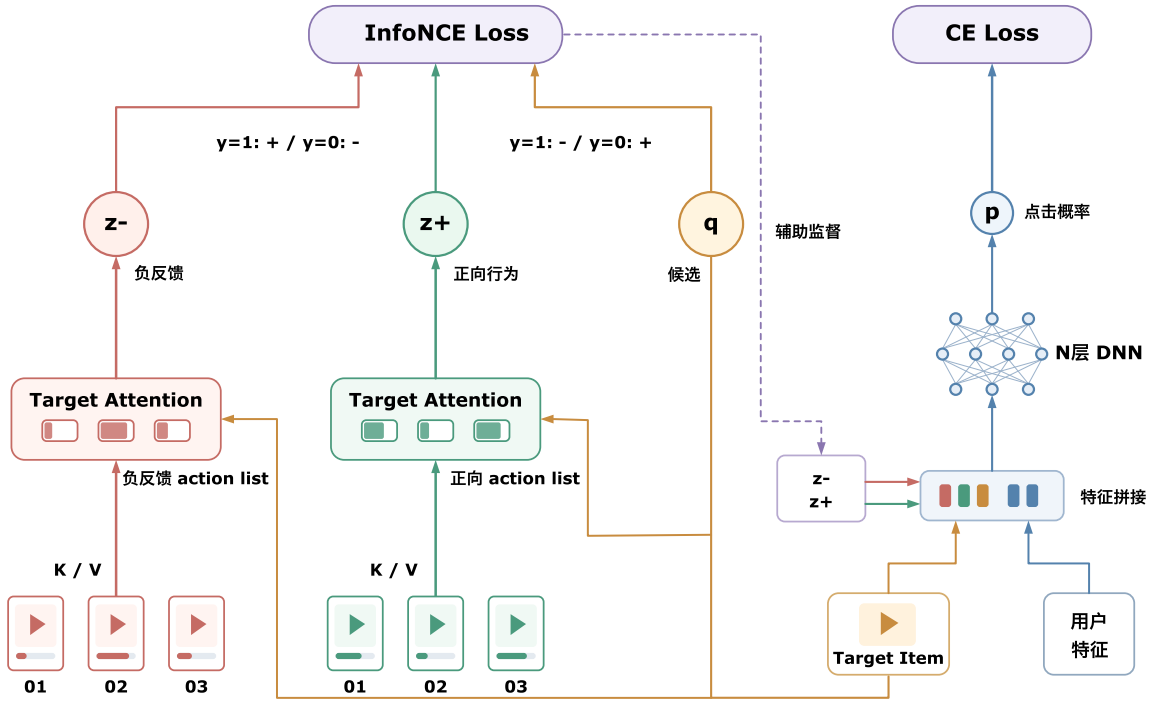


图 20.7: 基于正负反馈行为序列的对比学习用户行为序列建模。

如图20.7所示，候选视频特征在经过线性投影后，分别作为两路 Target Attention 模块的 Query，短播和非短播行为序列经过相应的 Embedding 层及线性变换后分别作为 Key 和 Value。通过两路独立的 Attention 计算，分别得到候选视频感知的负反馈表示 $\mathbf{z}^-$ 和正反馈表示 $\mathbf{z}^+$。上述两路 Attention 模块可以采用相同的网络结构，并根据具体实现选择共享或独立的参数。

与直接对历史行为 Embedding 进行平均池化或采用固定权重进行加权求和不同，Target Attention 能够根据当前候选视频动态调整不同历史行为的权重。同一用户的历史序列在面对不同候选视频时，可以提取出不同的正向和负向兴趣表示。随后，可以将 $\mathbf{z}^-$、$\mathbf{z}^+$ 与用户特征、候选视频特征以及上下文特征进行 Concat 拼接，输入 DNN 网络，最终输出候选视频是否点击等用户反馈的预测概率 $\hat{p}$。

在此基础上，我们还可以进一步构造对比学习辅助目标，使得模型能够识别当前候选视频更接近哪一类近期反馈。假设 $\mathbf{q}$ 表示候选视频的投影表示，$s(\mathbf{q}, \mathbf{z})$ 表示候选表示与用户兴趣表示之间的相似度函数，$y \in \{0, 1\}$ 表示当前候选曝光后的反馈标签，其中 $y = 0$ 表示发生短播，$y = 1$ 表示未发生短播。通过对两路兴趣表示构造二选一的 InfoNCE 损失：

$$\mathcal{L}_{\text{NCE}} = -y \log \frac{\exp(s(\mathbf{q}, \mathbf{z}^-)/\tau)}{\exp(s(\mathbf{q}, \mathbf{z}^-)/\tau) + \exp(s(\mathbf{q}, \mathbf{z}^+)/\tau)} - (1-y) \log \frac{\exp(s(\mathbf{q}, \mathbf{z}^+)/\tau)}{\exp(s(\mathbf{q}, \mathbf{z}^-)/\tau) + \exp(s(\mathbf{q}, \mathbf{z}^+)/\tau)} \tag{20.32}$$

其中，$\tau > 0$ 为温度系数，用于调节不同相似度之间的区分程度。当候选视频发生短播时，以负反馈表示 $\mathbf{z}^-$ 作为负样本表示，以正反馈表示 $\mathbf{z}^+$ 作为正样本表示。

需要注意的是，对比学习的目标并不是无条件地将候选视频推离所有短播历史，而是根据当前候选的真实反馈标签建立相应的正负样本关系。对于短播样本，模型应当学习候选视频与近期负反馈表示之间的关联；而对于非短播样本，则应当增强候选视频与正向兴趣表示之间的关联。因此，对比学习不是简单地将所有负反馈行为统一视为需要远离的内容，而是根据实际反馈动态调整兴趣表示与候选视频之间的相似度。

此外，当短播或非短播序列为空时，无法构造完整的二选一对比目标，因此需要对相应样本的辅助损失进

行掩码处理，仅利用两路序列均有效的样本计算对比损失。主预测任务则仍然可以使用现有可用特征进行训练。最终，以短播预测的交叉熵作为主损失，联合优化目标为：

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{NCE}} \tag{20.33}$$

其中，主任务损失定义为：

$$\mathcal{L}_{\text{CE}} = -y \log \hat{p} - (1-y)\log(1-\hat{p}) \tag{20.34}$$

λ 用于控制辅助对比学习目标的权重。在线推理时，模型只需要执行两路行为序列的编码与预测网络，不需要计算对比学习辅助损失损失计算的后续部分。在实际应用中，我们还需要通过线上 A/B 实验来评估模型对于短播率、观看时长、VV、多样性以及其他指标的影响，同时关注正负反馈行为序列的 Target Attention 计算所带来的额外开销与推理延迟。

总体来看，基于正负反馈行为序列的对比学习用户行为序列建模方法，通过分别构建短播与非短播行为序列，将用户近期的正向兴趣和负向反馈显式引入精排模型，并利用候选感知的注意力机制提取两类兴趣表示。在此基础上，对比学习进一步根据候选视频的真实反馈建立正负样本关系，为序列表示提供额外的区分性监督。这种方式能够补充传统正向预估任务对实时负反馈建模的不足，使得用户兴趣序列不仅能够反映用户喜欢什么，也能够反映用户近期对哪些内容表现出较低的兴趣。

20.1.6 基于增广行为序列的对比学习建模

除了直接利用正负反馈行为序列构造对比学习目标，还可以通过数据增强的方式，从同一用户的历史行为中构造多个不同视图，使得模型学习对合理行为扰动具有一定稳定性的用户兴趣表示。这种方法不需要改变精排模型原有的预测目标，而是在训练阶段引入额外的序列级监督，使模型能够进一步区分不同类型行为所反映的兴趣偏好。

在工业级推荐系统中，用户的历史行为序列通常包含不同强度和不同语义的反馈。例如，有效播放和长播通常能够反映用户的正向兴趣，而短播和主动 hate 行为（如用户明确标记不喜欢的视频）则可以作为负向反馈。通过对正向行为序列进行适当增广，并将其与负向行为序列进行对比，可以使得模型在保持正向兴趣表示稳定的同时，学习不同反馈类型之间的差异。

首先，按照行为发生时间，分别抽取用户最近至多 K 条有效播放、长播、短播和主动 hate 行为，记为 S^{eff}、S^{long}、S^{short} 和 S^{hate}。其中，有效播放和长播构成正向行为序列，短播和主动 hate 行为构成负反馈行为序列。每条行为记录可以包含视频 ID、播放时长以及其他行为属性，不同特征必须作为同一条行为记录整体参与序列的变换，以避免增广后视频 ID 与对应行为信息发生错位的问题。

对于有效播放和长播这两类正向序列，可以采用 Shuffle 操作构造多个增广视图。具体来说，将原始序列记为 x，在保持每条行为记录内部特征对应关系不变的情况下，对序列中的行为位置进行随机调整，得到 N 个正向增广视图 $x_1^+, \ldots, x_N^+$。短播和主动 hate 序列则作为负样本来源，进一步构造 $2N$ 个负向视图 $x_1^-, \ldots, x_{2N}^-$。注意这里的 N 表示每个锚点序列产生的正向增广视图数量，而不是原始行为序列的长度。Shuffle 的范围可以根据需要设置为整段序列或者局部的时间窗口。由于用户行为本身具有时间顺序，过强的随机打乱可能破坏近期兴趣和行为演化信息，因此在实际应用中需要限制 Shuffle 的范围，使得序列增广主要改变用户局部的行为顺序，而不是完全消除时序信息。

图20.8展示了基于增广行为序列的对比学习结构。对于每个正向行为序列，原始序列作为锚点，Shuffle 得到的多个增广序列作为正样本视图，而短播和主动 hate 序列经过相同的 Embedding 及序列表示映射后作为负样本视图。模型通过对比学习，使得同一正向行为序列及其不同增广视图在表示空间中保持接近，同时与负反馈序列保持区分。

假设 $f(\cdot)$ 表示将行为序列映射为定长表示的函数，$s(\mathbf{a}, \mathbf{b})$ 表示两个序列表示之间的相似度，通常采用余弦相似度，$\tau > 0$ 为温度系数。对于一个正向行为序列 x，将其原始表示 $f(x)$ 作为锚点，构造多正样本的对比学习损失：

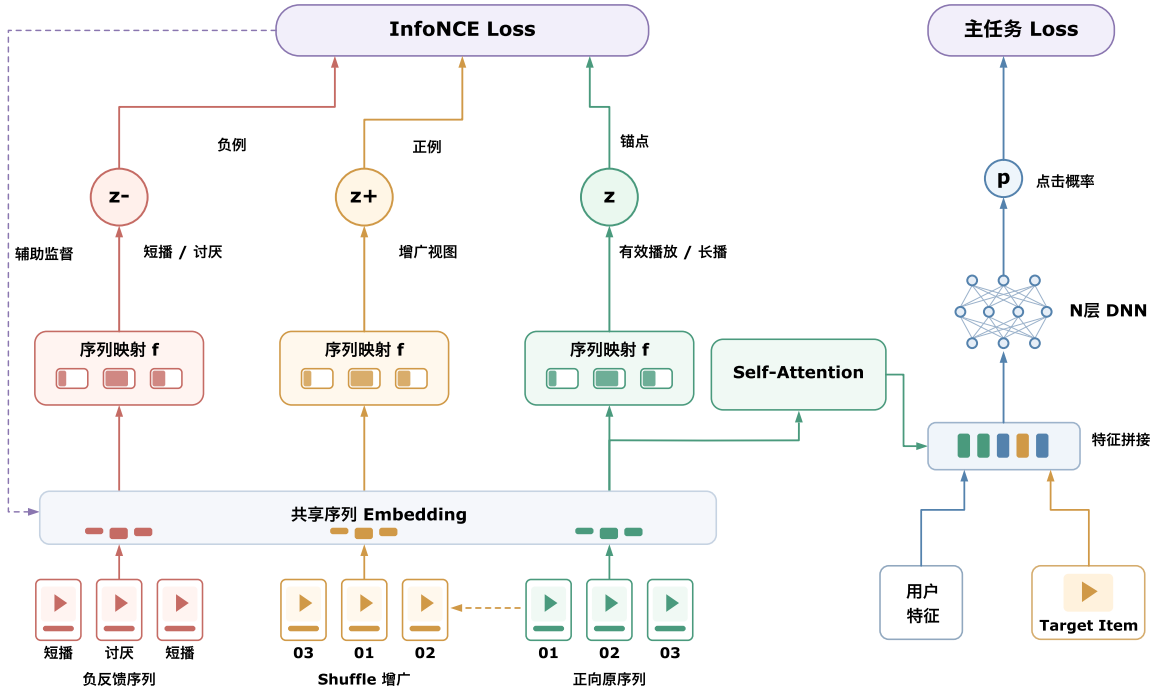


图 20.8: 基于正向行为序列增广的对比学习用户行为序列建模。

$$\mathcal{L}_{\text{aug}} = -\mathbb{E}_x \left[\log \frac{\sum_{i=1}^{N} \exp(s(f(x), f(x_i^+))/\tau)}{\sum_{i=1}^{N} \exp(s(f(x), f(x_i^+))/\tau) + \sum_{j=1}^{2N} \exp(s(f(x), f(x_j^-))/\tau)} \right] \tag{20.35}$$

其中，数学期望针对有效播放和长播两类可用的正向行为锚点进行计算。该目标将同一正向序列的多个增广视图共同作为正样本，并将短播和主动 hate 行为产生的视图作为负样本，使模型能够学习对正向行为序列增广具有一定稳定性的表示，同时保留正向兴趣与负向反馈之间的区分能力。

需要注意的是，Shuffle 本身只改变行为在序列中的位置，并不改变行为的语义类别，因此同一正向行为序列的不同 Shuffle 结果应当被视为正样本，而不能因为序列顺序发生变化就被视为不同的兴趣类别。另一方面，短播与主动 hate 行为也不一定与所有正向行为完全不相关，例如用户可能对某一类视频具有长期兴趣，但对某个具体视频产生短播或主动 hate。因此，构造负样本时需要结合行为时间、视频 ID 以及实际的业务语义来进行处理，避免将高度重叠或具有明显相同兴趣指向的行为错误地当作负样本。

此外，序列表示函数 f 也需要能够区分不同的行为顺序。如果直接使用完全不考虑位置的平均池化，那么对序列进行 Shuffle 前后的表示可能完全相同，数据增广也就难以提供有效的额外监督。因此，可以通过保留行为位置信息的序列映射来实现表示计算，并在增广过程中控制 Shuffle 的幅度，使其既能够产生不同的序列视图，又不会完全破坏行为序列原有的时序语义。

在模型结构方面，辅助对比学习分支与精排主模型的线上结构保持相对独立。在训练阶段，原始正向行为序列经过 Embedding 和 Self-Attention 网络来提取用户的兴趣表示，再与用户特征、候选视频特征及其他特征共同输入精排主网络，最终完成原有的点击率或排序预测任务。与此同时，正向增广视图和负向行为视图经过辅助序列映射函数 f，计算上述对比学习损失函数。

为了避免辅助对比学习改变主模型中已经建立的序列建模结构，可以对辅助分支的参数更新范围进行限制。辅助分支可以直接使用行为 Embedding，并采用固定的、能够保留位置差异的掩码来进行加权聚合，最终生成序列的表示。这样可以不引入额外的可训练序列编码层。通过这种方式，辅助的对比学习损失函数产生的梯度只会更新共享的行为 Embedding，不会流入主模型中的 Self-Attention、DNN 以及预测头等模块。而精排主任务的损失函数则仍然按照原有路径来更新行为 Embedding 和其他的模型参数。

最终，模型采用主任务损失与辅助对比学习损失进行联合优化：

$$\mathcal{L} = \mathcal{L}_{\text{base}} + \lambda \mathcal{L}_{\text{aug}} \tag{20.36}$$

其中，$\mathcal{L}_{\text{base}}$ 表示原有精排任务的损失，λ 用于控制辅助对比学习目标的权重。在线推理时，模型只保留原有正向行为序列的 Embedding 及 Self-Attention 建模路径，不再生成 Shuffle 增广视图，也不需要执行负反馈序列的辅助对比计算。这样也能够大幅降低线上推理的计算量，让对比学习模块只用于离线训练阶段。

总体来看，基于增广行为序列的对比学习方法通过对正向行为进行 Shuffle 构造多个语义相关的序列视图，并利用短播和主动 hate 行为构造负样本，为用户行为 Embedding 提供了额外的区分性监督。与前一种方法直接利用候选视频与正负反馈序列的匹配关系不同，该方法主要从序列表示学习的角度出发，来增强模型对于正向兴趣表示的稳定性，并使其与负反馈保持区分度。这两种方法分别从候选感知的反馈匹配和行为序列自身的表示学习两个角度利用对比学习方法，可以作为传统精排序列建模目标的补充。在实际应用时，我们通常需要结合主任务指标、正负样本构造方式及线上计算开销，对辅助损失的权重和行为增广强度进行系统性的评估。

20.2 基于 Attention 的用户行为序列建模

在工业级推荐系统中，用户的历史行为通常包含大量不同类型的交互记录，而这些行为对于当前推荐目标的参考价值并不相同。例如，用户近期观看过的视频可能反映其当前兴趣，但较早发生的行为也可能体现出相对稳定的长期偏好。与此同时，用户对不同候选物品的兴趣具有明显的目标相关性，同一段历史行为对于不同候选物品可能具有完全不同的参考价值。因此，如何根据当前候选物品，从用户历史行为中动态提取与推荐目标相关的兴趣信息，是用户行为序列建模需要解决的核心问题之一。

Attention 机制为这一问题提供了一种有效的解决思路。与传统的平均池化或固定权重聚合方式不同，Attention 能够根据当前候选物品与历史行为之间的相关性，为不同历史行为分配动态权重，并将加权后的行为表示作为用户兴趣特征输入后续推荐模型。这种目标感知的兴趣提取方式，使模型能够针对不同候选物品自适应地选择和利用用户历史行为中的有效信息。以 DIN 模型为代表的早期工作，将 Attention 机制引入 CTR 预估，通过局部兴趣激活机制实现候选物品感知的用户兴趣建模。

在此基础上，后续研究进一步将 Attention 机制与兴趣演化、会话建模以及时间信息相结合，不断增强模型对用户动态兴趣的刻画能力。例如，DIEN 通过兴趣提取与兴趣演化两个阶段建模用户兴趣随行为序列的变化；DSIN 从会话粒度组织用户历史行为，刻画不同会话中的兴趣及其演化关系；BST 则将 Transformer 中的 Self-Attention 引入推荐模型，建模历史行为之间的关联；TIN 进一步通过目标感知的时间编码与四元交互，联合捕获历史行为与目标物品之间的语义相关性和时序相关性。本节将沿着这一技术发展脉络，依次介绍上述代表性模型，并进一步讨论基于 Attention 的用户行为序列建模在处理超长历史行为时面临的问题，以及相应的改进方法。

20.2.1 DIN 模型

DIN（Deep Interest Network）模型是阿里巴巴提出的经典 CTR 预估模型，源自于 KDD 2018 会议的论文《Deep Interest Network for Click-Through Rate Prediction》[29]。该模型针对传统 Embedding + MLP 结构中用户兴趣表示缺乏个性化的问题，引入目标感知的 Attention 机制，根据当前候选广告与用户历史行为之间的相关性，动态提取用户的兴趣表示。

在传统的 Embedding + MLP 模型中，用户行为特征通常采用 Multi-hot 编码表示。由于不同用户的历史行为数量不同，Embedding 层得到的行为向量集合具有不同的长度，无法直接输入后续的全连接网络。因此，通常需要通过 Sum Pooling 或 Average Pooling 将多个行为 Embedding 聚合为一个定长向量，再与其他用户特征、广告特征和上下文特征进行拼接，输入 MLP 网络进行 CTR 预估。这种方式虽然能够将变长的行为序列转换为固定维度的特征表示，但所有历史行为都采用相同的聚合方式，没有充分考虑不同历史行为与当前候选广告之间的相关性。此外，用户往往具有多种不同的兴趣，而简单的 Pooling 会将这些兴趣压缩到同一个向量中，容易造成有价值的局部兴趣信息损失。

为了解决上述问题，DIN 模型引入了 Local Activation Unit，根据当前候选广告动态计算用户不同历史行为的激活权重，并通过加权求和得到候选广告感知的用户兴趣表示。DIN 模型的整体网络结构如图20.9所示，其中左图为传统的 Embedding + MLP 基线模型，右图为 DIN 模型。可以看出，DIN 模型保留了基线模型中的 Embedding、特征拼接和 MLP 等主要结构，核心改进在于对用户行为特征的聚合方式进行了重新设计。通过引入 Local Activation Unit，DIN 模型可以为每个历史行为计算与当前候选广告相关的激活权重，再对行为 Embedding 进行加权聚合。

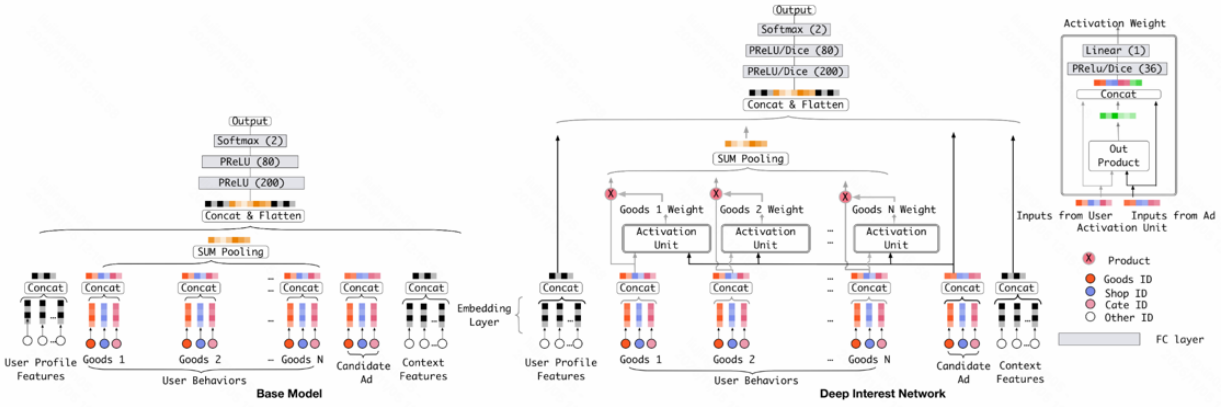


图 20.9: DIN 模型的网络结构（左：传统 Embedding + MLP 模型；右：DIN 模型）。

假设用户 U 的历史行为 Embedding 序列为 $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_H\}$，当前候选广告的 Embedding 为 $\mathbf{v}_A$，其中 H 表示用户历史行为的数量。Local Activation Unit 通过一个前馈神经网络计算候选广告与每个历史行为之间的相关性，得到对应的激活权重，并据此生成用户兴趣向量：

$$\mathbf{v}_U(A) = \sum_{j=1}^{H} a(\mathbf{e}_j, \mathbf{v}_A)\mathbf{e}_j = \sum_{j=1}^{H} w_j \mathbf{e}_j \tag{20.37}$$

其中，$a(\cdot)$ 表示 Local Activation Unit，w_j 表示第 j 个历史行为对应的激活权重。与传统 Pooling 得到的固定用户表示不同，$\mathbf{v}_U(A)$ 显式依赖于当前候选广告，因此同一个用户面对不同候选广告时，可以得到不同的兴趣表示，实现用户兴趣的个性化激活。

在 Local Activation Unit 的设计中，除了候选广告 Embedding 和历史行为 Embedding 两部分输入，DIN 模型还引入了两者之间的乘积交互特征，帮助网络显式建模行为与候选广告之间的相关性。与此同时，模型还采用了一种称为 Dice（Data Adaptive Activation Function）的自适应激活函数。Dice 可以看作 PReLU 激活函数的一种泛化，它根据输入数据的均值和方差动态调整激活函数的分界位置，而不是像传统 PReLU 一样固定在零点。Dice 的计算方式如下：

$$\begin{aligned} f(s) &= p(s)s + (1 - p(s))\alpha s \\ p(s) &= \frac{1}{1 + \exp\left(-\frac{s - \mathbb{E}[s]}{\sqrt{\mathrm{Var}[s] + \epsilon}}\right)} \end{aligned} \tag{20.38}$$

其中，α 为可学习参数，$\mathbb{E}[s]$ 和 $\mathrm{Var}[s]$ 分别表示当前输入的均值和方差，ϵ 为防止数值不稳定而引入的极小常数。在训练阶段，均值和方差通常根据当前的 Mini-batch 进行计算。而在推理阶段，上述均值和方差则使用训练过程中累计的滑动平均值。通过这种方式，Dice 激活函数能够根据输入数据的分布自适应地调整激活行为，提升模型对不同输入分布的适应能力。

此外，DIN 模型的 Attention 权重并没有采用传统 Softmax 进行归一化，即不要求所有历史行为的激活权重之和为 1。这一设计是为了保留用户兴趣的激活强度信息。当候选广告能够激活用户更多相关历史行为时，聚合结果的整体强度也可以相应增大；如果采用 Softmax 归一化，不同候选广告对应的兴趣表示会受到权重和为 1 的约束，从而损失这部分强度信息。因此，DIN 模型不仅能够根据候选广告选择相关历史行为，还能够在一定程度上反映用户对不同候选广告的兴趣强弱。

总的来说，DIN 模型的核心贡献并不是重新设计用户行为序列的编码结构，而是将传统的固定 Pooling 改进

为目标感知的 Attention Pooling，使用户兴趣表示能够随候选广告动态变化。其本质上提出了一种个性化的特征聚合方式，实现了用户兴趣的“千人千面”和“千物千面”。

不过，DIN 模型本身仍然采用基于加权求和的 Pooling 方式，并没有显式建模历史行为之间的时序关系。当用户历史行为序列较长时，即使引入了 Attention，模型仍需要对大量历史行为逐一计算激活权重，众多弱相关或噪声行为也可能参与最终的兴趣聚合，导致有效兴趣信息被稀释。在一些工业场景中，当行为序列长度增长到数百甚至更长时，Attention Pooling 的效果可能逐渐受到限制，实际收益甚至与 Average Pooling 相差不大。

此外，DIN 模型的加权聚合本身不保留行为发生的先后顺序，难以刻画用户兴趣随时间的动态演化。论文作者也尝试使用 LSTM 对用户历史行为进行序列建模，但实验中并未取得明显收益。作者认为，用户行为序列与自然语言文本不同，往往包含多个并行存在的兴趣，兴趣之间还可能频繁切换，因此直接采用传统循环网络并不能有效解决这一问题。因此，如何在保留目标感知兴趣提取能力的同时，进一步建模用户兴趣的动态演化，成为后续 DIEN 等模型重点研究的问题。

20.2.2 DIEN 模型

DIEN（Deep Interest Evolution Network）模型是阿里巴巴提出的经典 CTR 预估模型，源自于 AAAI 2019 会议的论文《Deep Interest Evolution Network for Click-Through Rate Prediction》[28]。该模型是在 DIN 模型的基础上进一步引入时序建模机制，可以捕捉用户兴趣随历史行为的不断变化。上一节提出的 DIN 模型在实际应用中通常存在以下问题：

笔记 DIN 模型通过 Attention 机制，根据当前候选广告与历史行为之间的相关性动态聚合用户兴趣，但其本质上仍然是一种加权 Pooling 操作，没有显式建模历史行为之间的时序依赖关系，也难以刻画用户兴趣的动态演化。

因此，DIEN 模型将用户兴趣建模划分为两个阶段：首先从历史行为序列中提取各个时间步的兴趣状态，然后结合当前候选广告，对相关兴趣的演化过程进行建模。

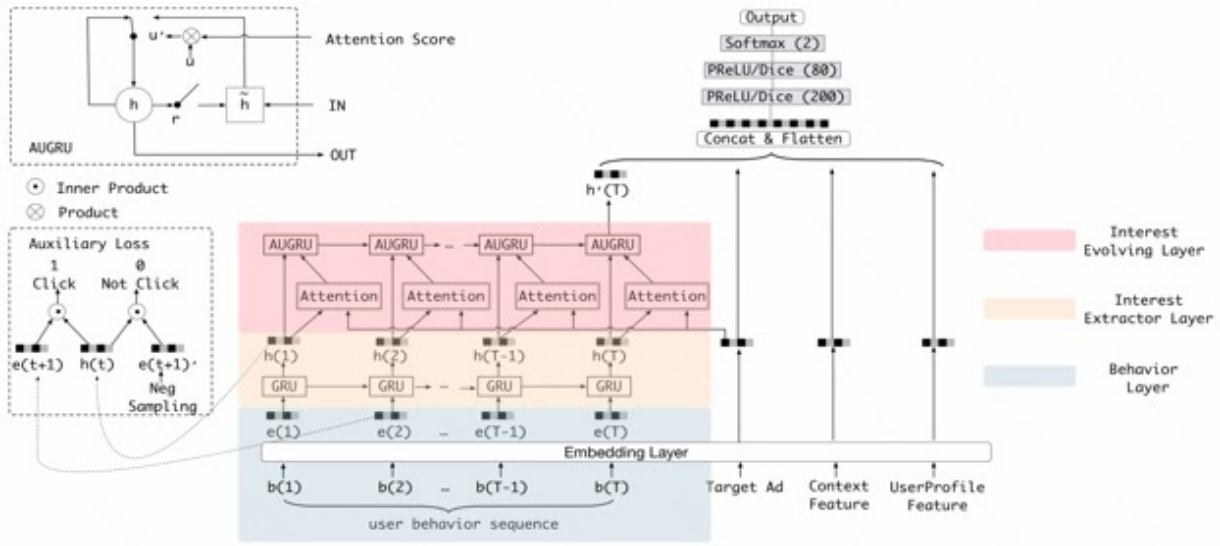


图 20.10: DIEN 模型的网络结构图。

DIEN 模型的整体网络结构如图20.10所示。模型首先将用户历史行为按照发生时间排序，并通过 Embedding 层将离散行为转换为稠密向量。随后，第一层 GRU 从历史行为中提取各个时间步的兴趣状态，并利用辅助损失对这些状态进行监督；第二层 GRU 则引入 Attention 机制，根据当前候选广告对不同兴趣状态进行激活，并进一步建模与候选广告相关的兴趣演化过程。最后，模型将得到的最终兴趣表示与用户画像、广告及上下文等特征拼接，输入 MLP 网络完成 CTR 预估。

在兴趣提取阶段，DIEN 模型使用 GRU 按照行为发生的时间顺序依次处理历史行为 Embedding，得到每个时间步对应的隐藏状态：

$$\mathbf{h}_t = \mathrm{GRU}(\mathbf{e}_t, \mathbf{h}_{t-1}) \tag{20.39}$$

其中 $\mathbf{e}_t$ 表示第 t 个历史行为的 Embedding，$\mathbf{h}_t$ 表示处理该行为之后得到的隐藏状态。与 DIN 模型直接对所有历史行为进行加权求和不同，GRU 网络能够结合此前的行为信息更新当前状态，从而将用户兴趣的变化过程编码到隐藏状态序列中。

不过，只依靠最终 CTR 预估损失进行训练时，监督信号主要来自于最终的预测结果，而序列中间各个时间步的隐藏状态缺少直接的监督。虽然这些隐藏状态会通过后续 GRU 参与最终兴趣表示的计算，但仅仅依靠最终预测目标进行反向传播，可能难以使每个隐藏状态都充分表达对应时刻的用户兴趣。

因此，DIEN 模型引入辅助损失函数（Auxiliary Loss），利用下一时刻的行为对当前隐藏状态进行额外监督。对于当前时刻的隐藏状态 $\mathbf{h}_t$，模型将用户下一时刻实际发生的行为 Embedding $\mathbf{e}_{t+1}$ 作为正样本，并从物品集合中采样未发生交互的物品作为负样本 $\hat{\mathbf{e}}_{t+1}$。通过判断隐藏状态与正负样本之间的匹配程度，使 $\mathbf{h}_t$ 能够更好地表达当前行为之后的用户兴趣。辅助损失的定义如下：

$$\mathcal{L}_{\text{aux}} = -\sum_{t=1}^{T-1} \left[\log \sigma(\mathbf{h}_t^\top \mathbf{e}_{t+1}) + \log(1 - \sigma(\mathbf{h}_t^\top \hat{\mathbf{e}}_{t+1})) \right] \tag{20.40}$$

其中，$\sigma(\cdot)$ 表示根据输入向量计算匹配概率的 Sigmoid 函数。最终，模型将辅助损失与 CTR 预估损失联合优化：

$$\mathcal{L} = \mathcal{L}_{\text{target}} + \alpha \mathcal{L}_{\text{aux}} \tag{20.41}$$

其中，$\mathcal{L}_{\text{target}}$ 为 CTR 预估任务的损失函数，α 为控制辅助损失权重的超参数。

通过引入辅助损失，DIEN 模型能够为序列中各个时间步的兴趣状态提供额外的监督信号，使 GRU 学习到更具表达能力的兴趣表示。同时，分布在整个行为序列上的辅助监督也能够缓解长序列训练时仅依靠最终预测目标进行反向传播所带来的优化困难。此外，下一行为预测任务还为 Embedding 层提供了额外的语义信息，有助于学习更有效的物品表示。经过这一阶段，模型得到完整的用户兴趣状态序列 $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T$，并将其作为后续兴趣演化建模的输入。

在获得兴趣状态序列之后，DIEN 模型进一步考虑用户兴趣随时间演化的特点。论文指出了用户兴趣演化主要具有两个特征。一方面，用户的兴趣可能在不同时期发生变化，即兴趣漂移（Interest Drift）。例如，用户可能在一段时间内集中浏览图书，而在另一段时间内更多关注服装。另一方面，不同类型的兴趣通常具有相对独立的演化趋势（Interest Individual），例如用户对图书和服装的兴趣变化并不完全同步。针对这两个特点，DIEN 模型将 Attention 的局部兴趣激活能力与 GRU 的时序建模能力结合起来，根据当前候选广告筛选相关的兴趣状态，并进一步建模这些兴趣随历史行为变化的过程。

DIEN 模型首先根据当前候选广告的 Embedding $\mathbf{e}_a$，计算每个兴趣状态 $\mathbf{h}_t$ 与候选广告之间的相关性，并通过 Softmax 得到对应的注意力权重：

$$a_t = \frac{\exp(\mathbf{h}_t^\top W \mathbf{e}_a)}{\sum_{j=1}^{T} \exp(\mathbf{h}_j^\top W \mathbf{e}_a)} \tag{20.42}$$

其中，W 为可学习的参数矩阵，a_t 表示当前候选广告对第 t 个兴趣状态的激活程度。注意力权重越大，说明该时刻的兴趣与当前候选广告越相关。在此基础上，DIEN 模型设计了三种将 Attention 与 GRU 结合的方式。

DIEN 模型的第一种方式称为 **AIGRU（GRU with Attentional Input）**，直接使用注意力权重对兴趣状态进行加权，并将加权后的结果作为第二层 GRU 的输入：

$$\mathbf{i}'_t = a_t \mathbf{h}_t \tag{20.43}$$

这种方式能够降低弱相关兴趣状态的输入强度，但由于 GRU 的隐藏状态还会受到前一时刻状态的影响，即使当前输入被缩小到零，也无法完全消除弱相关兴趣对状态更新的影响。

DIEN 模型的第二种方式称为 **AGRU（Attention based GRU）**，不再对 GRU 的输入进行加权，而是直接使用注意力权重替换原有的 Update Gate，控制隐藏状态的更新：

$$\mathbf{h}'_t = (1 - a_t)\mathbf{h}'_{t-1} + a_t \widetilde{\mathbf{h}}'_t \tag{20.44}$$

其中 $\widetilde{\mathbf{h}}'_t$ 表示 GRU 当前时刻的候选隐藏状态。通过这种方式，注意力权重能够直接控制当前兴趣状态对最终结果的影响。不过，AGRU 使用一个标量注意力权重替代了原本具有多个维度的 Update Gate，因而忽略了不同隐藏维度在状态更新过程中的重要性差异。

DIEN 模型的第三种方式称为 **AUGRU（GRU with Attentional Update Gate）**，也是 DIEN 最终采用的结构。该方法保留 GRU 原有的 Update Gate，并使用注意力权重对其进行调节，得到新的更新门控：

$$\widetilde{\mathbf{u}}_t' = a_t \mathbf{u}_t' \tag{20.45}$$

$$\mathbf{h}_t' = (1 - \widetilde{\mathbf{u}}_t') \odot \mathbf{h}_{t-1}' + \widetilde{\mathbf{u}}_t' \odot \widetilde{\mathbf{h}}_t' \tag{20.46}$$

其中，$\mathbf{u}_t'$ 为原始 GRU 的更新门控向量，$\widetilde{\mathbf{u}}_t'$ 为引入 Attention 后的更新门控，$\odot$ 表示逐元素相乘。与 AGRU 直接使用一个标量控制整个隐藏状态的更新不同，AUGRU 保留了原始 Update Gate 对不同隐藏维度的独立控制能力，同时利用注意力权重调节整体更新强度。当当前兴趣与候选广告的相关性较低时，注意力权重会相应减小更新门控，从而降低该兴趣状态对后续隐藏状态的影响。当相关性较高时，则允许模型更加充分地吸收当前兴趣信息。

相比起 AIGRU 和 AGRU，AUGRU 能够更好地兼顾目标感知的兴趣激活与 GRU 的状态更新机制，因此被 DIEN 模型选作最终的兴趣演化结构。经过第二层 GRU 的处理，最终隐藏状态 $\mathbf{h}_T'$ 作为与当前候选广告相关的用户兴趣表示，与其他用户、广告及上下文特征一起输入 MLP，完成最终 CTR 预估。总体而言，相比起 DIN 模型，DIEN 模型将用户兴趣建模从静态的加权聚合扩展为动态的兴趣提取与演化过程。一方面，第一层 GRU 结合辅助损失学习具有明确监督信号的兴趣状态序列；另一方面，第二层 GRU 通过 AUGRU 将候选广告感知的 Attention 引入状态更新过程，使模型能够重点关注与当前预测目标相关的兴趣演化轨迹。

不过，DIEN 模型仍然存在一定的局限性。首先，GRU 虽然能够按照行为发生顺序建模历史交互，但 DIEN 模型并没有显式考虑相邻行为之间的实际时间间隔。在工业级推荐场景中，用户历史行为可能集中发生在短时间内，也可能间隔数小时甚至数天，而 DIEN 模型主要依据行为发生的先后顺序进行状态更新。当两段行为序列具有相同的物品及其排列顺序，但相邻行为之间的时间间隔不同时，模型难以直接感知这种时间差异。其次，用户历史行为往往包含多个并行存在、频繁切换的兴趣，GRU 的递归状态更新容易受到不同兴趣之间切换的影响，难以始终保持各类兴趣的独立性。此外，尽管辅助损失能够为序列中间状态提供额外监督，但长序列上的循环计算仍然存在较高的计算开销和优化难度。如何进一步结合行为时间信息、兴趣之间的关联关系以及更高效的序列建模结构，仍然是后续研究需要解决的问题。

20.2.3 DSIN 模型

DSIN（Deep Session Interest Network）模型是阿里巴巴提出的 CTR 预估模型，源自 IJCAI 2019 会议论文《Deep Session Interest Network for Click-Through Rate Prediction》[9]。该模型在 DIEN 的基础上进一步考虑了用户行为在时间维度上的分布特点，将连续的历史行为划分为多个 Session，分别建模 Session 内部的兴趣关联以及不同 Session 之间的时序关系。DSIN 模型的基本观察如下：

笔记 同一个 Session 内用户的连续交互通常具有较强的相关性，反映了用户在一段时间内相对集中的兴趣；而不同 Session 之间可能存在较大的时间间隔，用户的兴趣也可能发生明显变化。因此，仅按照行为发生顺序对所有历史行为进行统一建模，难以充分刻画用户在不同时间段的兴趣特征。

DSIN 模型的整体网络结构如图20.11所示。在进入 MLP 之前，模型分为两条处理路径：一条将用户画像和目标物品等稀疏特征转换为对应的 Embedding 表示；另一条专门处理用户历史行为序列，从下到上依次包括 Session 划分、Session 兴趣提取、Session 兴趣交互和 Session 兴趣激活四个模块。首先，模型根据行为之间的时间间隔将历史序列划分为多个 Session；然后利用 Multi-head Self-Attention 提取每个 Session 内部的兴趣表示，并通过 Bi-LSTM 建模不同 Session 之间的时序关系；最后，模型根据当前目标物品对不同 Session 的兴趣表示进行动态加权。两路得到的特征经过拼接和展平之后，共同输入 MLP 网络完成最终的 CTR 预估。

在 Session 划分阶段，DSIN 模型根据相邻历史行为之间的时间间隔对用户行为序列进行分组。当相邻两次行为的时间间隔超过 30 分钟时，就将其视为不同 Session 之间的分界点。这种划分方式能够将时间上相对连续的行为组织在同一个 Session 中，使模型能够更集中地刻画用户在一段时间内的局部兴趣。

在完成 Session 划分之后，DSIN 模型需要分别提取每个 Session 内部的兴趣表示。由于同一个 Session 中的行为通常具有较强的相关性，DSIN 模型使用 Multi-head Self-Attention 建模 Session 内不同历史行为之间的关联，从

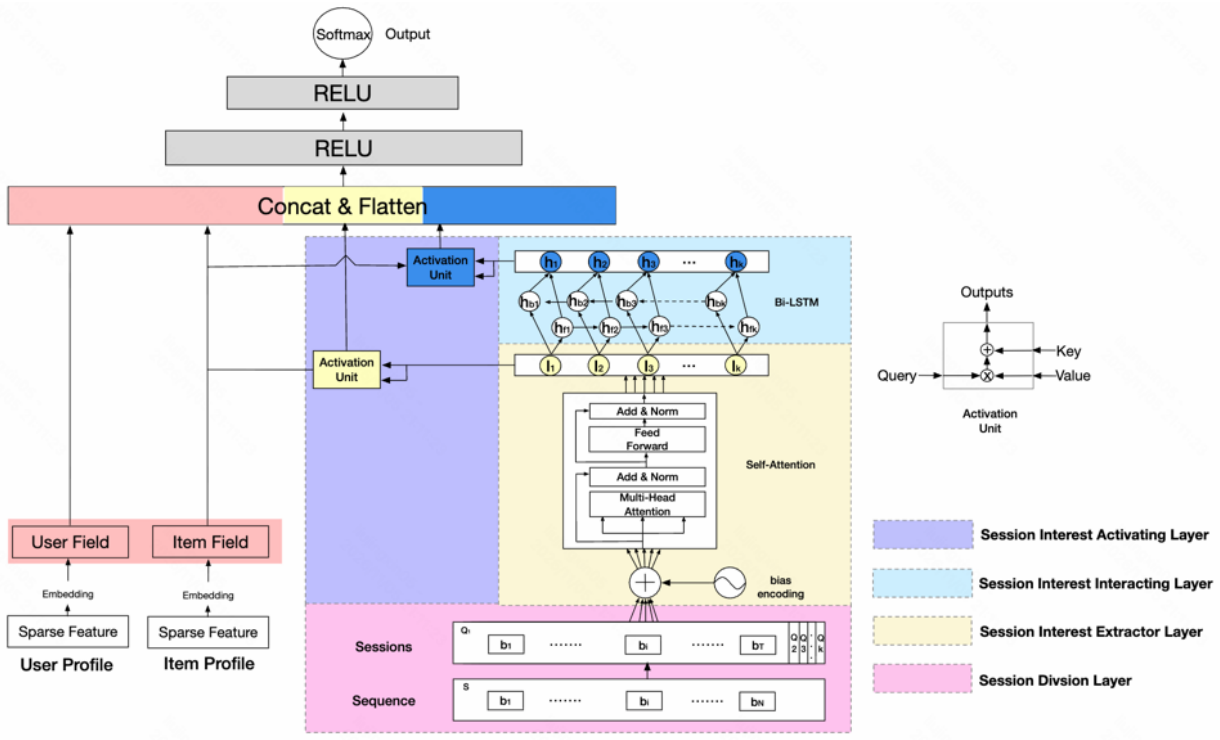


图 20.11: DSIN 模型的网络结构图。

而提取更加充分的 Session 兴趣表示。与传统 Self-Attention 不同，DSIN 模型还引入了偏置编码（Bias Encoding），在行为 Embedding 的基础上增加可学习的偏置编码：

$$BE(k,t,c) = w_k^K + w_t^T + w_c^C \tag{20.47}$$

其中，k 表示 Session 的索引，t 表示行为在当前 Session 中的位置，c 表示行为 Embedding 的维度，w^K、w^T 和 w^C 分别为对应的可学习偏置参数。与 Transformer 中根据位置预先构造的位置编码（Position Encoding）不同，偏置编码直接通过模型来学习不同 Session、行为位置以及 Embedding 维度上的偏置。

将偏置编码加入行为 Embedding 之后，DSIN 模型可以得到带偏置的表示 $\mathbf{Q}$：

$$\mathbf{Q}_k = \mathbf{E}_k + \mathbf{BE}_k \tag{20.48}$$

然后 DSIN 模型再通过 Multi-head Self-Attention（MHSA）捕获同一个 Session 内不同历史行为之间的关联，并经过前馈网络、残差连接和 Layer Normalization 得到各行为的输出表示：

$$\begin{aligned} \mathbf{Q}' &= [\mathbf{Q}_{k1}, \cdots, \mathbf{Q}_{kH}] \\ \mathbf{I}_k^Q &= \text{LayerNorm}(\text{FFN}(\text{MHSA}(\mathbf{Q}')) + \mathbf{Q}') \end{aligned} \tag{20.49}$$

其中 H 是 head 的总个数。最后，DSIN 模型对同一个 Session 内所有行为的表示进行平均池化，得到对应的 Session 兴趣向量 $\mathbf{I}_k$：

$$\mathbf{I}_k = \frac{1}{H}\sum_{i=1}^{H} \mathbf{I}_{k,i}^Q \tag{20.50}$$

由于不同 Session 共享 Self-Attention 的网络参数，所以 DSIN 模型能够以相同的特征提取方式学习到不同时间段的用户兴趣。

在获得各个 Session 的兴趣表示之后，DSIN 模型进一步考虑不同 Session 之间的时序关系。虽然不同 Session 可能对应不同的兴趣主题，但这些兴趣并不是完全孤立的，其变化仍然具有一定的上下文关联。例如，用户在不同时段对不同品类物品的关注程度可能受到此前兴趣状态的影响。因此，DSIN 模型使用 Bi-LSTM 对 Session

兴趣序列进行建模，将前向和后向两个方向的隐藏状态进行拼接，得到融合上下文信息的 Session 表示：

$$\begin{aligned}\mathbf{h}_k &= \text{Bi-LSTM}(\mathbf{I}_k) \\ \mathbf{H}_k &= \overrightarrow{\mathbf{h}}_k \oplus \overleftarrow{\mathbf{h}}_k\end{aligned} \tag{20.51}$$

其中，$\mathbf{I}_k$ 为 Bi-LSTM 的输入，$\mathbf{H}_k$ 为融合前后文信息后的输出表示，$\oplus$ 表示向量拼接。与直接对所有历史行为使用 RNN 建模不同，Bi-LSTM 网络处理的是经过 Session 内兴趣提取之后的向量序列，因此其建模粒度由单个行为转变为 Session，更加关注不同时间段用户兴趣之间的关联及其变化。

最后，为了从多个 Session 中提取与当前目标物品相关的兴趣，DSIN 模型引入 Target Attention 机制，分别对原始 Session 兴趣表示 $\mathbf{I}_k$ 和融合上下文后的表示 $\mathbf{H}_k$ 进行加权聚合。具体来说，DSIN 模型首先根据目标物品 Embedding $\mathbf{x}_I$ 计算不同 Session 兴趣与目标物品之间的相关性：

$$a_k^I = \frac{\exp(\mathbf{I}_k^\top W_I \mathbf{x}_I)}{\sum_{j=1}^K \exp(\mathbf{I}_j^\top W_I \mathbf{x}_I)} \tag{20.52}$$

$$a_k^H = \frac{\exp(\mathbf{H}_k^\top W_H \mathbf{x}_I)}{\sum_{j=1}^K \exp(\mathbf{H}_j^\top W_H \mathbf{x}_I)} \tag{20.53}$$

其中，K 为 Session 的数量，W_I 和 W_H 为可学习的参数矩阵。随后，分别对两组 Session 表示进行加权求和：

$$\mathbf{U}_I = \sum_{k=1}^K a_k^I \mathbf{I}_k, \qquad \mathbf{U}_H = \sum_{k=1}^K a_k^H \mathbf{H}_k \tag{20.54}$$

其中，$\mathbf{U}_I$ 表示基于原始 Session 兴趣表示得到的用户兴趣向量，$\mathbf{U}_H$ 则表示融合不同 Session 上下文信息后的用户兴趣向量。通过分别聚合这两路表示，模型既能够保留各个 Session 本身所表达的兴趣特征，也能够利用不同 Session 之间的时序关联信息。最终，$\mathbf{U}_I$、$\mathbf{U}_H$ 与用户画像、目标物品等特征的 Embedding 进行拼接和展平，输入 MLP 完成 CTR 预估。

总的来说，DSIN 模型将用户行为序列的建模粒度从单个行为扩展到了 Session，通过 Multi-head Self-Attention 建模 Session 内部的行为关联，再利用 Bi-LSTM 捕获不同 Session 之间的时序关系，最后结合目标感知的 Attention 提取与当前物品相关的兴趣表示。相比起 DIEN 模型直接使用 GRU 来对所有历史行为进行连续建模，DSIN 模型通过显式的 Session 划分，将用户兴趣的局部集中性与跨时间段的变化结合起来，为用户行为序列提供了更加分层的建模方式。

20.2.4 BST 模型

BST（Behavior Sequence Transformer）模型是阿里巴巴提出的基于 Transformer 的用户行为序列建模模型，源自 DLP-KDD 2019 会议论文《Behavior Sequence Transformer for E-commerce Recommendation in Alibaba》[4]。该模型将 Transformer 引入工业 CTR 预估任务，利用 Self-Attention 建模用户历史行为之间的关联，从而获得更加充分的行为序列表示。与 DIEN 模型通过 GRU 逐步提取兴趣状态、DSIN 模型先划分 Session 再进行分层建模不同，BST 模型直接对用户历史行为序列进行 Self-Attention 运算，使每个行为都能够根据序列中其他行为的信息更新自身表示。

BST 模型的整体网络结构如图20.12所示。模型整体采用 Embedding + Transformer + MLP 的结构。首先，BST 模型将用户画像、上下文、目标物品及交叉特征等输入 Embedding 层，得到对应的稠密表示；对于用户历史行为序列，则同时利用物品 ID、类目 ID 和位置特征构造行为 Embedding，并将目标物品一并输入 Transformer。经过 Multi-head Self-Attention 和前馈网络后，BST 模型得到融合历史行为关联信息的序列表示，并取目标物品对应的输出向量作为当前用户的目标感知兴趣表示。最后，BST 模型将该表示与其他特征 Embedding 拼接，输入三层的 MLP 网络，通过 Sigmoid 激活函数最终输出 CTR 的预估结果。

在特征表示阶段，BST 模型将输入特征分为两部分：一部分是用户画像、上下文、目标物品以及交叉特征等其他特征（Other Features），另一部分是用于构建用户行为序列的物品特征。对于每个历史行为物品，模型主要采用物品 ID 和类目 ID 作为序列特征，并将位置特征融入对应的行为表示。

此外，BST 模型也引入了位置编码。与原始 Transformer 使用固定的正弦和余弦函数构造位置编码不同，BST

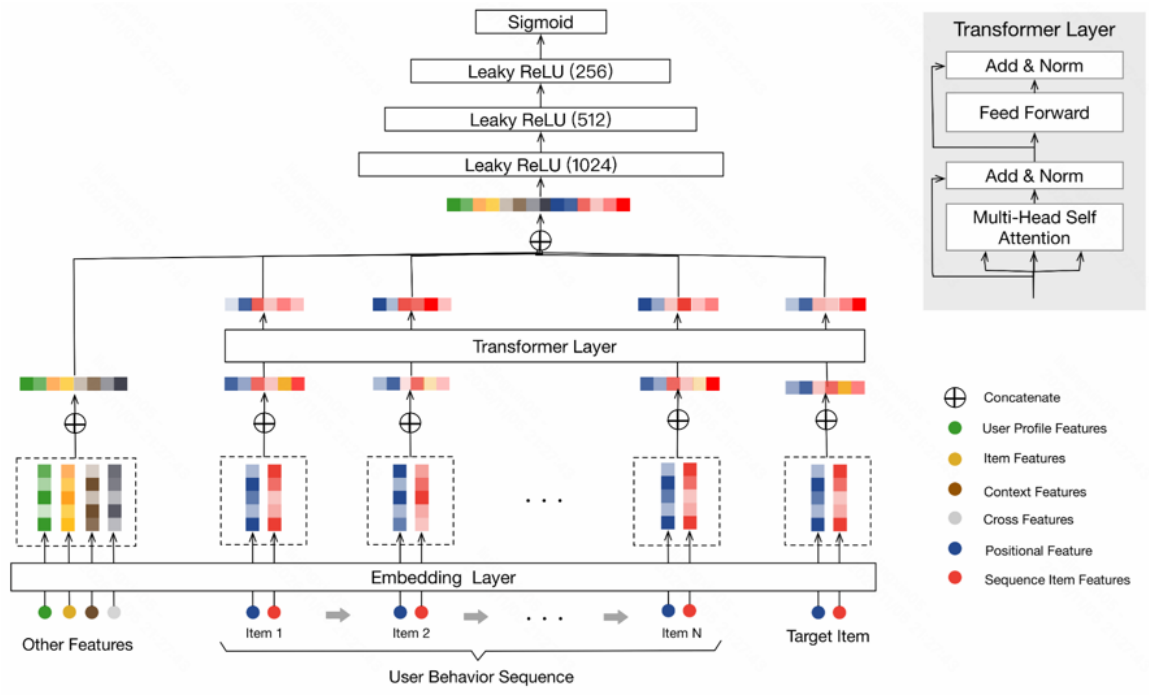


图 20.12: BST 模型的网络结构图。

模型根据历史行为发生时间与当前推荐时间之间的间隔构造位置特征。对于第 i 个历史行为物品 v_i，其位置特征定义为：

$$\text{pos}(v_i) = t(v_{\text{target}}) - t(v_i) \tag{20.55}$$

其中，$t(v_{\text{target}})$ 表示当前推荐请求对应的时间，$t(v_i)$ 表示历史行为发生的时间。这种位置编码的方式在工业界也被称为 **timediff 位置编码**。通过这种方式，BST 模型不仅能够区分不同行为在序列中的位置，还能够利用行为距离当前推荐时刻的时间间隔信息，使 Transformer 在学习行为关联时具备一定的时间感知能力。最终，历史行为及目标物品经过 Embedding 处理后，作为 Transformer 的输入。

在 Transformer 建模阶段，BST 使用 Multi-head Self-Attention 捕获用户历史行为之间的关联。模型首先通过不同的线性映射得到 Query、Key 和 Value，然后利用 Scaled Dot-Product Attention 计算不同物品之间的注意力权重：

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V} \tag{20.56}$$

其中，d 为 Query 和 Key 的特征维度。为了从不同的特征子空间中学习行为之间的关联，BST 模型进一步采用 Multi-head Attention，将多个注意力头的输出进行拼接和线性变换：

$$\text{MHA}(\mathbf{E}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}^H \tag{20.57}$$

其中，$\mathbf{E}$ 表示输入的行为序列 Embedding，h 为注意力头的数量，$\mathbf{W}^H$ 为输出映射矩阵。经过 Multi-head Self-Attention 后，模型还使用逐位置前馈网络（Point-wise Feed-Forward Network）进一步提取非线性特征，并通过残差连接、Layer Normalization 和 Dropout 改善模型的训练稳定性与泛化能力。多个这样的 Transformer Block 可以进行堆叠，以进一步学习行为序列中的高阶关联。不过，通过实验验证发现，BST 模型使用一个 Transformer Block 的效果优于堆叠两层或三层的结构。因此 BST 模型最终采用单层 Transformer 以兼顾模型效果和计算效率。

在得到 Transformer 的输出之后，BST 模型并不会对所有历史行为的输出向量进行简单的平均或求和，而是取目标物品对应的输出表示。由于 Transformer 的 Self-Attention 已经对输入序列中的不同物品进行了信息交互，因此目标物品对应的输出向量不仅包含其自身的特征信息，还融合了用户历史行为中的相关信息，能够作为当前候选物品感知的用户兴趣表示。最后，模型将该向量与前面得到的用户画像、上下文、目标物品及交叉特征等 Embedding 拼接，输入三层全连接网络，并通过 Sigmoid 得到最终的点击概率。

总的来说，BST 模型的核心改进在于利用 Transformer 的 Self-Attention 直接建模用户历史行为之间的关联，使模型不再局限于 GRU 的逐步状态更新，也不需要像 DSIN 模型一样显式划分 Session。与此同时，通过将历史行为与当前推荐时间之间的间隔作为位置特征，BST 模型在一定程度上结合了行为顺序与时间信息。但随着用户历史行为序列不断增长，Self-Attention 需要计算不同历史行为之间的两两关联，其计算复杂度和显存开销会随序列长度呈平方级增长。在工业级推荐场景中，直接对超长行为序列进行全量 Self-Attention 建模仍然面临较大的计算压力，这也推动了后续研究从行为筛选、序列压缩以及高效注意力机制等方向进一步优化长序列建模。

20.2.5 TIN 模型

TIN（Temporal Interest Network）模型是腾讯提出的用户兴趣建模方法，源自 WWW 2024 会议的论文《Temporal Interest Network for User Response Prediction》[30]。该模型针对已有用户兴趣模型难以同时捕获历史行为与目标物品之间语义相关性和时间相关性的问题，提出了目标感知的时间编码和显式交互机制，联合建模行为与目标之间的语义—时间相关性（Semantic-Temporal Correlation）。

在用户行为序列中，历史行为与当前目标物品之间通常同时存在语义和时间上的关联。例如，当用户点击一个服装类目下的目标物品时，历史上浏览过的同类物品通常具有更高的参考价值。而在这些相关行为中，近期发生的行为往往又比很久以前的行为更能反映用户当前的兴趣。作者通过互信息对这种语义—时间相关性进行了分析，发现历史行为与目标物品属于相同类别时，二者之间的相关性通常更强，并且这种相关性会随着历史行为距离目标物品的时间位置变远而逐渐衰减。作者进一步分析了 DIN、SASRec 和 BST 等模型，发现它们都难以充分捕获这两种相关性的联合关系。例如，DIN 模型缺乏显式的时间信息，而 SASRec 和 BST 模型虽然引入了时序编码，但对于行为与目标之间的语义相关性仍然建模不足。

针对这一问题，TIN 模型在传统 Embedding + MLP 结构的基础上设计了 Temporal Interest Module（TIM），专门用于建模用户历史行为与目标物品之间的语义—时间相关性。TIN 模型的整体网络结构如图20.13所示。模型首先将用户侧、物品侧和上下文侧的特征转换为 Embedding 表示，然后将用户历史行为及当前目标物品输入 TIM。该模块通过目标感知的时间编码，为行为和目标物品分别引入相对时间信息，并结合目标感知的 Attention 和目标感知的表示进行显式交互，最终得到用户兴趣向量。最后，将该向量与其他特征 Embedding 拼接，输入 MLP 完成 CTR 预估。

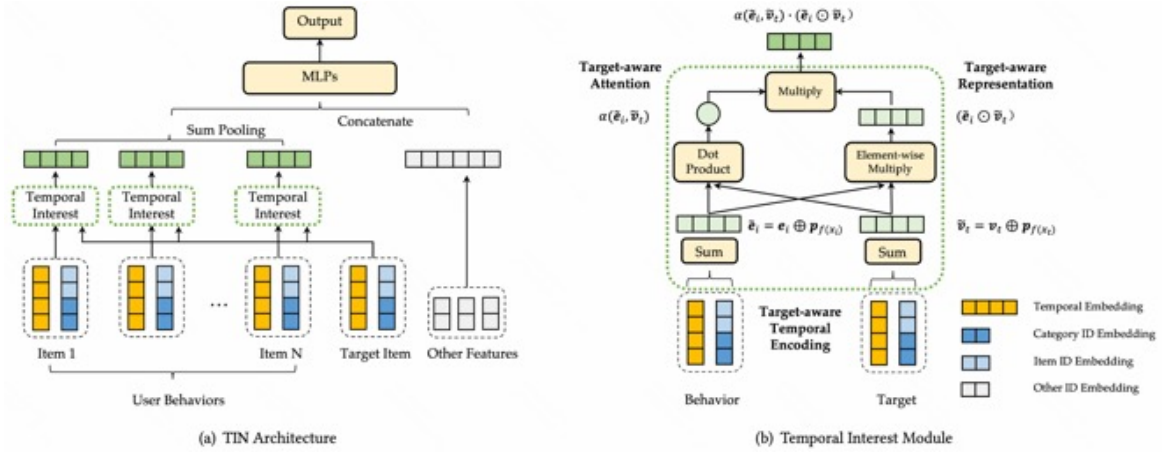


图 20.13: TIN 模型的网络结构图。

TIN 模型的核心是 TIM，它的关键在于同时考虑行为和目标物品的语义表示及其相对时间信息。首先，模型通过目标感知的时间编码（Target-aware Temporal Encoding，TTE）为历史行为和目标物品引入时间信息。与 BST 等模型按照历史行为发生的先后顺序进行位置编码不同，TTE 以当前目标物品为时间参考点，直接描述每个历史行为相对于目标物品的时间位置。

TIN 模型的时间编码总共有两种实现方式。第一种是基于相对位置的时间编码（TTE-P）。假设将用户历史

行为和目标物品按照发生时间排列为

$$\{X_1, X_2, \ldots, X_H, X_t\} \tag{20.58}$$

其中 H 为历史行为数量，X_t 表示当前目标物品。对于第 i 个历史行为，其相对位置定义为：

$$f_{\text{TTE-P}}(X_i) = H - i + 1 \tag{20.59}$$

而目标物品本身的位置定义为零。模型根据该相对位置查找对应的时间 Embedding，从而表示历史行为距离当前目标物品的相对远近。

TIN 模型的第二种时间编码是基于时间间隔的时间编码（TTE-T）。该方法直接利用历史行为与当前目标物品之间的实际时间差构造时间特征。第 i 个历史行为与目标物品之间的时间间隔为：

$$\tau_i = TS_t - TS_i \tag{20.60}$$

其中，TS_i 和 TS_t 分别表示历史行为和目标物品对应的时间戳。考虑到时间间隔是连续数值，模型首先通过分桶函数 $b(\cdot)$ 将其离散化，再查找对应的时间 Embedding：

$$\mathbf{p}_f(X_i) = \mathbf{p}_{b(\tau_i)} \tag{20.61}$$

其中，$b(\cdot)$ 可以采用等频分桶等方式实现。与 TTE-P 类似，目标物品本身也作为时间参考点，其对应的时间间隔为零。通过上述两种方式，模型可以根据具体场景选择使用相对位置或者实际时间间隔来描述历史行为的时间信息。

在获得时间 Embedding 后，TIN 模型将其与原有的物品语义 Embedding 逐元素相加，形成同时包含语义和时间信息的行为表示与目标表示：

$$\widetilde{\mathbf{e}}_i = \mathbf{e}_i + \mathbf{p}_f(X_i), \qquad \widetilde{\mathbf{v}}_t = \mathbf{v}_t + \mathbf{p}_f(X_t) \tag{20.62}$$

其中，$\mathbf{e}_i$ 和 $\mathbf{v}_t$ 分别表示第 i 个历史行为和目标物品的语义 Embedding。经过这一处理，历史行为和目标物品的表示同时包含了物品本身的语义信息以及相对于当前推荐目标的时间信息，为后续的联合交互提供了基础。

在获得上述表示后，TIN 模型进一步设计了目标感知的 Attention（Target-aware Attention，TA）和目标感知的表示（Target-aware Representation，TR），分别刻画历史行为与目标物品之间的相关性和显式交互。首先，目标感知的 Attention 使用 Scaled Dot-Product 计算每个历史行为与目标物品之间的相关性，并在所有历史行为上进行 Softmax 归一化：

$$\alpha(\widetilde{\mathbf{e}}_i, \widetilde{\mathbf{v}}_t) = \text{softmax}_i\left(\frac{\widetilde{\mathbf{e}}_i^\top \widetilde{\mathbf{v}}_t}{\sqrt{d}}\right) \tag{20.63}$$

其中，d 为 Embedding 维度，$\alpha(\cdot)$ 表示第 i 个历史行为对应的注意力权重。与 DIN 模型中的 Local Activation Unit 类似，该权重用于衡量不同历史行为对当前目标的贡献程度，但 TIN 模型在计算注意力时已经同时考虑了行为与目标的语义和时间信息。

当然，只通过 Attention 对历史行为进行加权，还不足以充分保留行为与目标之间的显式交互信息。因此，TIN 模型进一步引入目标感知的表示，对每个历史行为表示与目标物品表示进行逐元素相乘：

$$\mathbf{r}_i = \widetilde{\mathbf{e}}_i \odot \widetilde{\mathbf{v}}_t \tag{20.64}$$

其中，$\odot$ 表示逐元素相乘。与直接将加权后的行为 Embedding 输入 MLP 不同，这种方式显式保留了行为和目标物品在各个 Embedding 维度上的交互信息。最终，TIN 模型将 Attention 权重与目标感知的表示结合，并对所有历史行为进行加权聚合，得到最终的用户兴趣表示：

$$\mathbf{u}_{\text{TIN}} = \sum_{i \in H} \alpha(\widetilde{\mathbf{e}}_i, \widetilde{\mathbf{v}}_t)\,(\widetilde{\mathbf{e}}_i \odot \widetilde{\mathbf{v}}_t) \tag{20.65}$$

其中 H 表示用户的历史行为数量。这里有一个值得注意的设计，即：

笔记 TIN 模型中的 Attention 权重本身已经结合行为和目标的语义—时间信息，而逐元素交互进一步显式建模了二者在表示空间中的匹配关系。两者相乘后，TIN 模型实际上在行为语义、目标语义、行为时间和目标时间四个维度上构建了显式的四元交互（4-way Interaction），从而更充分地捕获历史行为与当前目标之间的语义—时间相

关性。

最终，TIN 模型得到的用户兴趣表示 $\mathbf{u}_{\text{TIN}}$ 与其他用户、物品及上下文特征拼接后输入 MLP，通过 Sigmoid 输出用户响应概率，整个模型采用交叉熵损失进行优化。总的来说，TIN 模型的核心改进在于将目标感知的时间编码与显式的二阶特征交互结合起来，不再仅依赖 Attention 权重进行历史行为筛选，而是同时利用注意力分配和行为—目标交互表示提取用户兴趣。相比起 DIN 模型缺少显式时间信息，以及 DIEN、DSIN 和 BST 等模型主要依靠 Attention 加权行为表示再通过 MLP 学习特征组合的方式，TIN 模型进一步显式构造了目标感知的二阶交互，使模型能够更充分地学习行为与目标之间的语义和时间关系。

20.2.6 Attention 序列建模的问题与解决方案

前面介绍的 DIN、DIEN、DSIN、BST 和 TIN 等模型，都通过 Attention 机制从用户历史行为中提取与当前候选物品相关的兴趣信息。然而，在工业推荐系统的长期行为建模场景中，Attention 机制并不一定能够充分利用所有历史行为。随着模型训练的持续进行，注意力权重可能逐渐集中到少数几个历史行为上，而其他行为的权重则接近于零。这种现象被称为 **Attention Polarization（注意力极化）**。当模型过度依赖少数历史行为时，用户多样化的兴趣难以得到充分表达，也会影响整个行为序列表示的学习效果。

快手科技在 WWW 2024 会议中发表的论文《De-Anchor: Mitigating Attention Polarization for Lifelong User Behavior Modeling in Click-Through Rate Prediction》[16] 对这一问题进行了专门研究。论文发现，在基于 Multi-Head Target Attention 的用户终身行为序列建模中，Attention 权重容易逐渐呈现接近 One-hot 的分布，即单个或少数几个历史行为获得绝大部分注意力权重，而其他行为几乎不参与最终的兴趣聚合。作者认为，这一现象与模型对用户个性化特征的过度依赖有关。随着用户训练样本不断积累，用户 ID 等个性化特征的 Embedding 能够逐渐学习到稳定的用户表示，模型对历史行为序列的依赖随之降低。然而，Softmax 归一化要求所有历史行为的注意力权重之和为 1，即使当前行为序列提供的信息并不充分，模型仍然需要将注意力分配给某些历史行为，最终可能使少数行为获得不必要的高权重，并在训练过程中不断强化这种偏向。

为了量化 Attention 的极化程度，作者设计了两个度量指标：Anchor Rate 和 Attention Entropy。对于用户 u 的一次样本，假设其历史行为序列长度为 T，Attention Head 的数量为 h，$w_{j,t}$ 表示第 j 个 Attention Head 对第 t 个历史行为分配的注意力权重。Anchor Rate 用于衡量不同 Attention Head 中是否存在权重接近 1 的历史行为，其定义为：

$$R_{u,\epsilon} = \frac{1}{h}\sum_{j=1}^{h}\mathbb{I}\left(\max_{1\le t\le T} w_{j,t} > 1-\epsilon\right) \tag{20.66}$$

其中，ϵ 为控制极化判定阈值的超参数，$\mathbb{I}(\cdot)$ 为指示函数。当某个 Attention Head 中存在一个历史行为获得接近 1 的注意力权重时，该 Head 被认为出现了极化现象。Anchor Rate 表示发生极化的 Attention Head 所占比例，取值范围为 $[0,1]$。另一个指标 Attention Entropy 用于衡量注意力权重分布的集中程度：

$$E_u = -\frac{1}{h}\sum_{j=1}^{h}\sum_{t=1}^{T} w_{j,t}\log w_{j,t} \tag{20.67}$$

当注意力权重趋向于均匀分布时，Attention Entropy 较大。而当权重逐渐集中到某一个历史行为时，Entropy 则趋近于零。因此，较高的 Anchor Rate 和较低的 Attention Entropy 通常意味着更加严重的注意力极化。论文在 Taobao 数据集上的实验发现，高活跃用户具有更高的 Anchor Rate 和更低的 Attention Entropy。在进一步过滤掉这些获得极高注意力权重的历史行为后，模型在高活跃用户上的 GAUC 并没有明显下降。这说明部分极化行为实际上并没有为预测提供足够有效的信息，模型对用户个性化特征的过度依赖可能是造成这一现象的重要原因。

针对上述问题，De-Anchor 模型从模型优化和 Attention 权重分配两个方面提出了改进方法。首先，Anchor-aware Gradient Dropout（AGD）通过动态丢弃用户个性化特征的部分梯度，降低模型对用户唯一表示的过度依赖。AGD 会根据用户的 Anchor Rate 调整用户 Embedding 的梯度丢弃概率：

$$p_{u,\epsilon,\tau} = \sqrt{R_{u,\epsilon}}\cdot\left(1-\mathbb{I}(N_u \le \tau)\right) \tag{20.68}$$

其中，N_u 表示用户 u 已经参与训练的样本数量，τ 为控制梯度丢弃启用条件的阈值。当用户的训练样本数量不

超过 τ 时，不执行梯度丢弃；当 $N_u > \tau$ 时，才根据 $\sqrt{R_{u,\epsilon}}$ 决定用户个性化 Embedding 的梯度丢弃概率。也就是说，用户的 Attention 极化程度越高，其个性化 Embedding 的梯度被丢弃的概率也越大。通过这种方式，模型在部分训练样本上跳过用户唯一 Embedding 的梯度更新，从而减少对个性化记忆特征的依赖，促使模型更加充分地利用共享的历史行为特征。同时，对于训练样本较少的用户，模型仍然保留正常的 Embedding 更新，以避免影响冷启动用户的特征学习。

另一种方法是 Target-aware Attention Anchor（TAA），它直接针对 Softmax 归一化造成的注意力权重过度集中问题进行改进。传统 Target Attention 只在真实历史行为之间分配注意力权重，所有行为的权重之和必须为 1。为了避免这一限制，De-Anchor 在历史行为序列前额外加入一个可学习的伪 Anchor 向量，并将其与真实历史行为一起参与 Attention 的 Softmax 计算。对于第 t 个历史行为，其注意力权重可以表示为：

$$\alpha_t = \frac{\exp(s_t)}{\exp(s_{\text{anchor}}) + \sum_{j=1}^{T} \exp(s_j)} \tag{20.69}$$

其中，s_t 表示当前目标与第 t 个历史行为之间的注意力得分，s_{anchor} 表示当前目标与伪 Anchor 之间的注意力得分。对于标准的 Scaled Dot-Product Attention，这两个得分分别由历史行为 Key 或 Anchor Key 与目标 Query 的缩放点积计算得到。伪 Anchor 本身是一个可学习向量，不对应任何真实用户行为，其作用是为 Attention 权重提供额外的分配位置，吸收那些不需要分配给真实历史行为的注意力权重。因此，真实历史行为的注意力权重之和不再被限制为 1：

$$\sum_{t=1}^{T} \alpha_t = \frac{\sum_{t=1}^{T} \exp(s_t)}{\exp(s_{\text{anchor}}) + \sum_{t=1}^{T} \exp(s_t)} \leq 1 \tag{20.70}$$

通过上述引入伪 Anchor 的机制，De-Anchor 模型不再需要将全部注意力权重强制分配给真实历史行为，而是可以根据当前目标与历史行为的相关性，自适应地决定实际参与兴趣聚合的信息量。这种设计既保留了真实历史行为之间的相对注意力关系，又为模型提供了抑制不必要注意力分配的空间，从而缓解注意力极化现象。

总的来说，De-Anchor 模型从模型优化和 Attention 权重分配两个方面缓解了用户终身行为序列建模中的注意力极化问题。AGD 通过限制用户唯一 Embedding 的梯度更新，降低模型对个性化记忆特征的过度依赖；TAA 则通过引入可学习的伪 Anchor，为 Attention 权重提供额外的分配空间，避免历史行为之间被迫竞争全部注意力。两种方法分别作用于特征学习和注意力计算过程，可以作为插件应用于不同的 CTR 预估模型，在保留原有序列建模结构的基础上，改善用户多样化历史兴趣的利用效果。

20.3 基于 Memory 的用户行为序列建模

前面介绍的 DIN、DIEN、DSIN 和 BST 等模型，主要通过 Attention 或 RNN 从用户历史行为中提取兴趣表示。这些方法通常需要在实时预估过程中读取并处理用户历史行为序列，随着序列长度不断增加，模型的计算开销和存储成本也会随之增长。对于具有数百甚至上千条历史行为的用户，如何在充分利用长期行为信息的同时，满足工业推荐系统对在线预估延迟和吞吐量的要求，成为一个重要的工程问题。基于 Memory 的用户行为序列建模方法通过引入固定容量的外部记忆单元，将用户历史行为逐步编码为紧凑的兴趣状态，使模型能够在不需要每次重新处理完整历史序列的情况下，持续更新和利用用户兴趣。本节将介绍这一方向的代表性工作 MIMN 模型和 LMN 模型，以及它们在算法设计与在线服务架构上的联合优化思路。

20.3.1 MIMN 模型

MIMN（Multi-channel User Interest Memory Network）模型是阿里巴巴提出的工业级长序列用户兴趣建模模型，源自于 KDD 2019 会议的论文《Practice on Long Sequential User Behavior Modeling for Click-Through Rate Prediction》[19]。该方法较早在工业级 CTR 预估场景中实现了千级长度用户历史行为序列的建模。它的核心思想如下：

定义 20.3

MIMN 模型借鉴了 Neural Turing Machine（NTM）的外部记忆机制，将用户长期行为信息持续写入固定大小的 Memory，并通过多个兴趣通道提取和维护用户的不同兴趣表示。与此同时，MIMN 模型并没有仅从模型结构上解决长序列建模问题，而是进一步设计了 User Interest Center（UIC），将用户兴趣的更新从实时预估请求中解耦出来，实现算法与线上服务架构的联合优化。

在工业级推荐系统中，用户历史行为通常占据特征集合的绝大部分。在阿里巴巴的广告系统中，用户行为特征约占全部特征的 90%。虽然增加历史行为长度能够带来明显的预估效果提升，但直接处理超长序列会面临存储和计算两方面的挑战。一方面，用户历史行为通常需要存储在高性能的分布式内存系统中，以便实时预估服务能够快速读取。随着行为序列长度增加，存储开销也会近似线性增长。

以论文中的线上系统为例，对于超过 6 亿用户，最大行为序列长度为 150 时，相关存储开销约为 1 TB；当序列长度增加到 1000 时，存储需求将增长到约 6 TB。另一方面，Attention 和 RNN 等序列模型需要在实时请求到达后处理历史行为，序列越长，计算开销越大。论文实验表明，在每个 Worker 的 QPS 为 500 时，DIEN 模型处理长度为 150 的行为序列已经需要约 14 ms。而当序列长度增加到 1000 时，模型的推理延迟会进一步增长到约 200 ms，远超线上系统的延迟要求。

针对上述问题，MIMN 模型从线上服务架构和用户兴趣建模算法两个方面进行改进。首先，在服务架构层面，论文提出了 User Interest Center（UIC），专门负责维护和更新用户的兴趣表示。传统的实时预估系统通常在收到推荐请求后，读取用户历史行为，并在预测服务内部完成兴趣建模，随后将兴趣表示与其他特征一起输入 CTR 模型。无论采用 Attention 还是 RNN，这部分计算都需要占用实时请求的处理时间。引入 UIC 之后，用户兴趣的计算被独立部署到单独的服务中，其更新不再依赖推荐请求，而是由用户实时产生的新行为事件触发。UIC 根据新增行为增量更新用户兴趣状态，并将最新的兴趣表示保存到在线存储系统中。这样，实时预估服务只需要读取已经计算好的用户兴趣表示，而不必在每次请求到达时重新处理完整的历史行为序列。

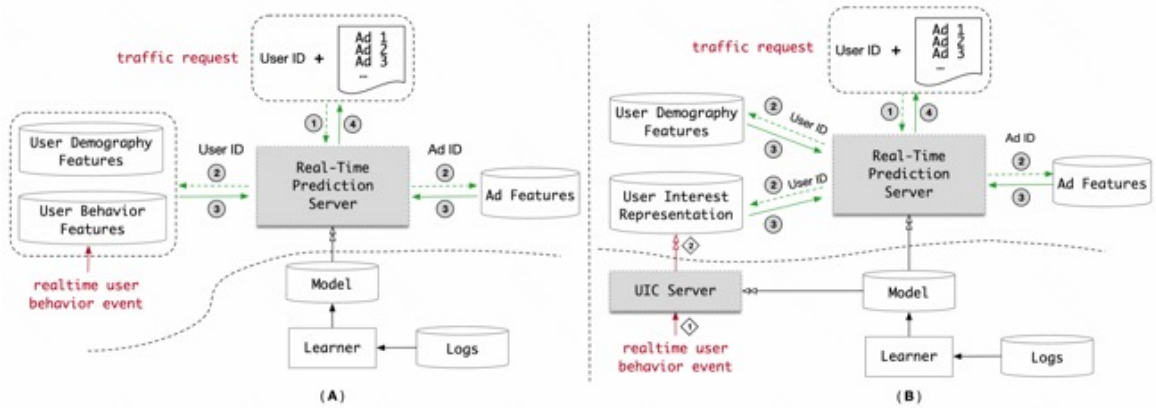


图 20.14: 引入 UIC 前后的实时预估服务架构对比。

图20.14展示了引入 UIC 前后实时预估系统的变化。在传统架构中，用户兴趣表示由 Prediction Server 在收到请求后进行计算。而在引入 UIC 后，兴趣建模被独立到 UIC Server 中，并由用户行为事件触发更新。Prediction Server 只需要读取 UIC 维护的最新兴趣表示，再与当前请求中的其他特征一起完成 CTR 预估即可。通过这种解耦式的设计，实时请求的处理时间不再被用户兴趣计算所占用，从而将耗时较大的用户兴趣计算从实时请求链路中分离出来。

在 MIMN 模型的线上实现中，UIC 可以将 DIEN 模型在长度为 1000、QPS 为 500 的场景下的推理延迟从约 200ms 降低到约 19ms，使得超长行为序列能够在原有实时预估系统中得到实际应用。不过，UIC 主要解决的是实时计算延迟问题，如何在有限的存储空间中充分保留用户长期行为所包含的兴趣信息，仍然需要从模型结构上进行优化。

因此，MIMN 模型借鉴 Neural Turing Machine（NTM）的设计，引入固定大小的外部 Memory，将用户历史行为逐步编码到多个记忆槽（Memory Slot）中。与直接存储完整行为序列不同，Memory 通过持续的读写操作维

护用户的历史兴趣状态，使其存储空间不再随历史行为数量线性增长。MIMN 模型的整体网络结构如图20.15所示，模型主要包含两部分：左侧负责用户兴趣建模，采用 NTM Memory 和 Memory Induction Unit（MIU）处理历史行为序列；右侧则延续传统的 Embedding + MLP 结构，将左侧提取的用户兴趣表示与其他用户及物品特征进行拼接，完成最终的 CTR 预估。因此，MIMN 模型的主要创新集中在左侧的记忆网络设计中。

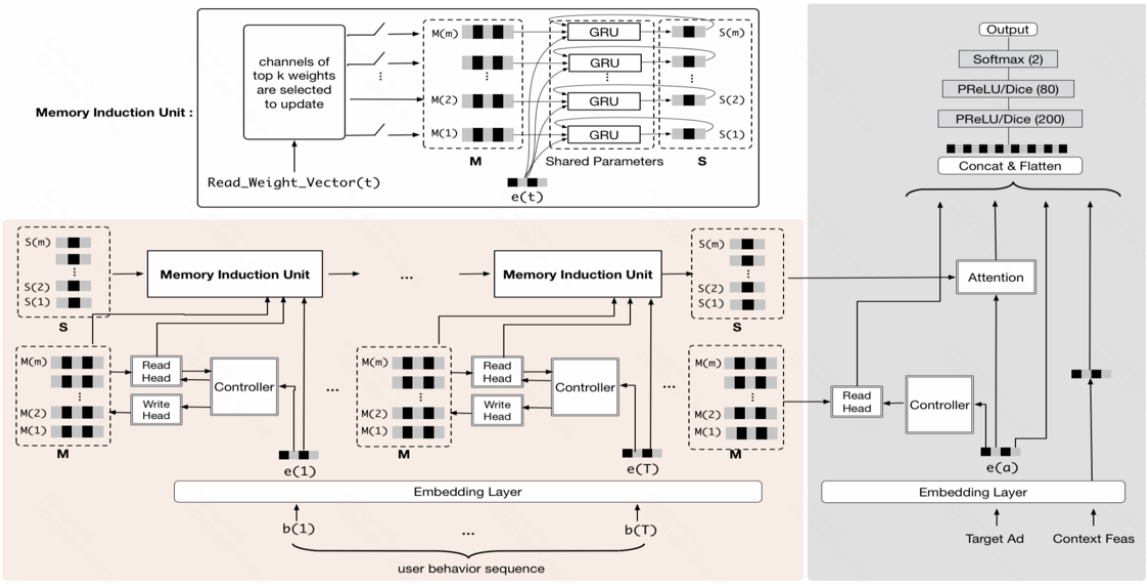


图 20.15: MIMN 模型的整体网络结构。

这里我们对 NTM 的记忆机制进行简要说明。在 NTM 中，外部 Memory 由 m 个记忆槽组成，在处理第 t 个行为时，记忆状态表示为：

$$\mathbf{M}_t = [\mathbf{M}_t(1), \mathbf{M}_t(2), \ldots, \mathbf{M}_t(m)] \tag{20.71}$$

其中，$\mathbf{M}_t(i)$ 表示第 i 个 Memory Slot 中存储的信息。NTM 通过一个控制器（Controller）根据当前输入的行为 Embedding 生成读写操作所需的向量，并通过 Memory Read 和 Memory Write 两个过程实现信息的提取与更新。

对于 Memory Read，控制器首先根据当前行为生成读取键向量 $\mathbf{k}_t$，再计算该向量与各个 Memory Slot 的相似度，得到读取权重：

$$w_t^r(i) = \frac{\exp\left(\text{sim}(\mathbf{k}_t, \mathbf{M}_{t-1}(i))\right)}{\sum_{j=1}^{m} \exp\left(\text{sim}(\mathbf{k}_t, \mathbf{M}_{t-1}(j))\right)} \tag{20.72}$$

其中，$\text{sim}(\cdot,\cdot)$ 表示余弦相似度。随后，根据读取权重对所有 Memory Slot 进行加权求和，得到当前行为对应的记忆读取结果：

$$\mathbf{r}_t = \sum_{i=1}^{m} w_t^r(i)\mathbf{M}_{t-1}(i) \tag{20.73}$$

对于 Memory Write，控制器以类似的方式生成写入权重 $\mathbf{w}_t^w$，同时生成 Add Vector 和 Erase Vector，分别表示需要写入和擦除的信息。Memory 根据写入权重更新对应的记忆槽，其更新过程可以表示为：

$$\mathbf{M}_t = (1 - \mathbf{E}_t) \odot \mathbf{M}_{t-1} + \mathbf{A}_t \tag{20.74}$$

其中，

$$\mathbf{E}_t = \mathbf{w}_t^w \otimes \mathbf{e}_t^{\text{erase}}, \qquad \mathbf{A}_t = \mathbf{w}_t^w \otimes \mathbf{a}_t \tag{20.75}$$

$\mathbf{e}_t^{\text{erase}}$ 和 $\mathbf{a}_t$ 分别表示擦除向量和写入向量，$\otimes$ 表示外积，$\odot$ 表示逐元素相乘。通过这样的读写机制，NTM 能够根据当前行为与已有记忆内容之间的相关性，将新的信息写入合适的 Memory Slot，并在后续行为到来时继续读取和更新。

不过，直接使用标准 NTM 还存在一个问题，即不同 Memory Slot 的使用可能不够均衡。在用户兴趣建模场景中，热门物品更容易出现在用户的历史行为中，如果这些行为频繁访问或更新少数几个 Memory Slot，就可能

导致部分记忆槽被过度使用，而其他记忆槽长期得不到充分利用。这样不仅浪费了有限的记忆空间，也会降低模型对用户多样化兴趣的存储能力。

针对这一问题，MIMN 模型引入了 Memory Utilization Regularization，通过对不同 Memory Slot 的累计写入权重进行约束，促使记忆资源得到更加均衡的利用。模型首先将第 t 个时间步之前各个记忆槽的历史写入权重进行累计：

$$\mathbf{g}_t = \sum_{c=1}^{t} \widetilde{\mathbf{w}}_c^w \tag{20.76}$$

然后根据累计利用情况计算权重转移矩阵：

$$\mathbf{P}_t = \text{softmax}(\mathbf{W}_g \mathbf{g}_t) \tag{20.77}$$

并对当前写入权重进行重新分配：

$$\widetilde{\mathbf{w}}_t^w = \mathbf{w}_t^w \mathbf{P}_t \tag{20.78}$$

其中，$\mathbf{W}_g$ 为可学习参数，$\widetilde{\mathbf{w}}_t^w$ 表示经过调整后的写入权重。对整个行为序列的写入权重进行累加，可以得到各个 Memory Slot 的总使用量：

$$\widetilde{\mathbf{w}}^w = \sum_{t=1}^{T} \widetilde{\mathbf{w}}_t^w \tag{20.79}$$

在此基础上，Memory Utilization Regularization 通过惩罚不同 Memory Slot 总写入权重的方差，鼓励各个记忆槽得到更加均衡的利用：

$$\mathcal{L}_{\text{reg}} = \lambda \sum_{i=1}^{M} \left(\widetilde{\mathbf{w}}^w(i) - \frac{1}{M} \sum_{j=1}^{M} \widetilde{\mathbf{w}}^w(j) \right)^2 \tag{20.80}$$

其中，λ 为正则化系数，M 是 Memory Slot 的数量。

需要注意的是，MIMN 模型并没有直接采用传统的 Least Recently Used（LRU）策略来控制记忆槽的使用。LRU 倾向于在相邻时间步更新不同的 Memory Slot。但在用户兴趣建模场景中，同一类兴趣可能连续出现多次，将相关行为写入相同的记忆槽反而有利于保持兴趣信息的连续性。因此，MIMN 模型通过可学习的权重调整和正则化实现整体使用均衡，而不是强制每个时间步切换记忆槽。

除了利用 NTM 存储历史行为信息，MIMN 模型还进一步设计了 Memory Induction Unit（MIU），用于提取记忆中的高阶兴趣信息。NTM 的主要目标是尽可能将原始行为信息存储到固定大小的外部 Memory 中，但仅依靠记忆读写还不足以充分建模不同用户兴趣自身的动态演化过程。例如，用户在一段时间内可能持续表现出对某一类物品的兴趣，这种兴趣随行为逐步变化的过程并不只是简单地保存历史行为信息，还需要对不同兴趣状态进行持续更新。

因此，MIU 在 NTM 的基础上增加了一组内部记忆 $\mathbf{S}_t$，同样包含 M 个 Memory Slot，并将每个 Slot 视为一个独立的用户兴趣通道（Interest Channel）。在每个时间步，MIU 首先根据 NTM 的读取权重 $\mathbf{w}_t^r$，选择权重最高的 k 个兴趣通道，仅对这些通道进行更新。对于被选中的第 i 个兴趣通道，其状态通过 GRU 进行更新：

$$\mathbf{S}_t(i) = \text{GRU}\left(\mathbf{S}_{t-1}(i), \mathbf{M}_t(i), \mathbf{e}_t\right) \tag{20.81}$$

其中，$\mathbf{S}_{t-1}(i)$ 为该兴趣通道上一时刻的状态，$\mathbf{M}_t(i)$ 为 NTM 中对应的记忆内容，$\mathbf{e}_t$ 为当前行为 Embedding。通过同时利用外部 Memory 中已经积累的历史信息和当前行为输入，MIU 能够在不同兴趣通道中持续更新用户兴趣状态。不同通道共享 GRU 参数，因此不会因为增加兴趣通道而显著增加模型参数量。这样，NTM 主要负责存储和提取原始行为信息，而 MIU 则进一步在这些信息的基础上建模高阶兴趣特征及其演化过程。

在在线服务中，MIMN 模型与 UIC 服务进一步结合，实现了用户兴趣的增量计算。图20.16展示了用户兴趣建模子网络在 UIC Server 中的部署方式。NTM 和 MIU 均被部署在 UIC 中，负责处理用户新产生的行为事件，并增量更新对应的外部 Memory 和多通道兴趣状态。当新的用户行为到达时，UIC 不需要重新读取和处理用户的全部历史行为，而是利用已有记忆状态完成一次增量更新，并将最新的用户兴趣表示写入在线存储系统。当实时预估服务收到请求后，只需要读取对应用户已经更新好的兴趣表示，再结合当前请求中的目标物品和其他特

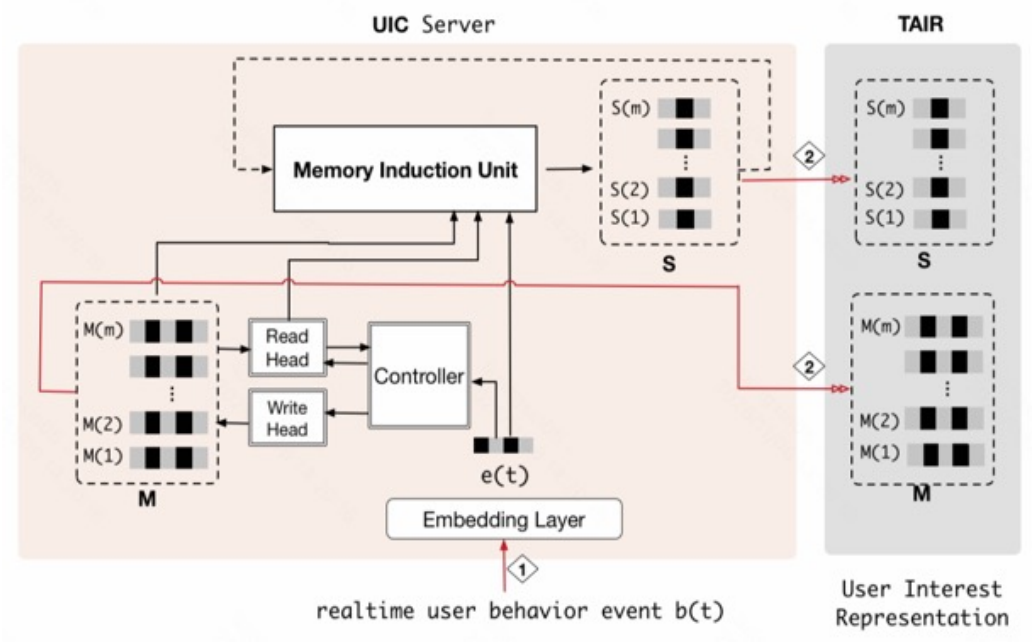


图 20.16: MIMN 用户兴趣建模子网络在 UIC 中的部署。

征进行预测。

通过 UIC 服务与 MIMN 模型的联合设计，用户历史行为不再需要以完整原始序列的形式参与每次实时预估，而是逐步压缩为固定大小的记忆状态，使得模型能够持续处理不断增长的用户行为序列。在线上系统中，这套框架设计不仅能够显著降低用户行为序列的存储压力，同时也减少了实时计算的延迟和资源消耗。这表明 MIMN 模型的价值不仅在于改进了长序列兴趣表示的学习方式，更在于通过模型与服务架构协同设计，使得千级用户行为序列能够在工业级 CTR 预估系统中得到应用。从处理机制来看，UIC 可以持续接收任意数量的新行为并更新固定大小的记忆状态，因此不再受到实时读取完整历史序列长度的直接限制。

尽管 MIMN 能够显著降低超长行为序列的存储和在线计算开销，但其记忆压缩机制仍然存在一定的局限性。首先，随着用户历史行为长度进一步增长，固定大小的 Memory 需要承载越来越多的历史信息，不同兴趣及其演化过程可能被压缩到有限的记忆槽中，从而引入信息混淆和噪声，难以始终准确地提取与当前推荐目标相关的兴趣。其次，MIMN 模型的记忆更新主要由用户历史行为驱动，维护的是相对独立于候选物品的用户兴趣表示，并没有像 DIN 和 DIEN 模型一样，在兴趣提取过程中显式引入当前目标物品。因此，MIMN 模型的 Memory 缺少目标感知能力，无法针对不同候选物品动态调整记忆读取过程。同一个用户的记忆表示需要服务于不同的候选物品，这在一定程度上限制了模型对细粒度目标相关兴趣的提取能力。

20.3.2 LMN 模型

LMN（Large Memory Network）模型是字节跳动提出的大规模记忆网络模型，源自 WWW 2025 会议的论文《Large Memory Network for Recommendation》[17]。该模型针对传统推荐模型中用户历史行为信息存储容量有限、长期兴趣表示不充分的问题，引入一个由所有用户共享的大规模 Memory，通过用户和物品特征生成查询，从记忆空间中检索与当前行为相关的信息。与 MIMN 模型为每个用户维护独立的兴趣记忆不同，LMN 模型将 Memory 设计为跨用户共享的参数空间，使得不同用户的行为信息能够在同一记忆空间中进行存储和复用，从而在时间维度上积累用户长期兴趣信息，并在用户之间实现兴趣知识的共享。

LMN 模型的整体结构如图20.17所示，主要包含两个阶段，即 Memory Extraction 和 Memory Injection。前者根据用户历史行为和用户特征生成查询，从共享 Memory 中检索 Top-K 个相关记忆槽，并将检索结果作为额外的兴趣信息；后者则通过辅助记忆损失，将用户与物品交互信息注入 Memory，使其能够持续学习和存储历史交互知识。整个模型仍然采用传统的 Embedding + MLP 结构，将 Memory 输出与其他用户、物品及交叉特征结合，最终完成 CTR 的预估。

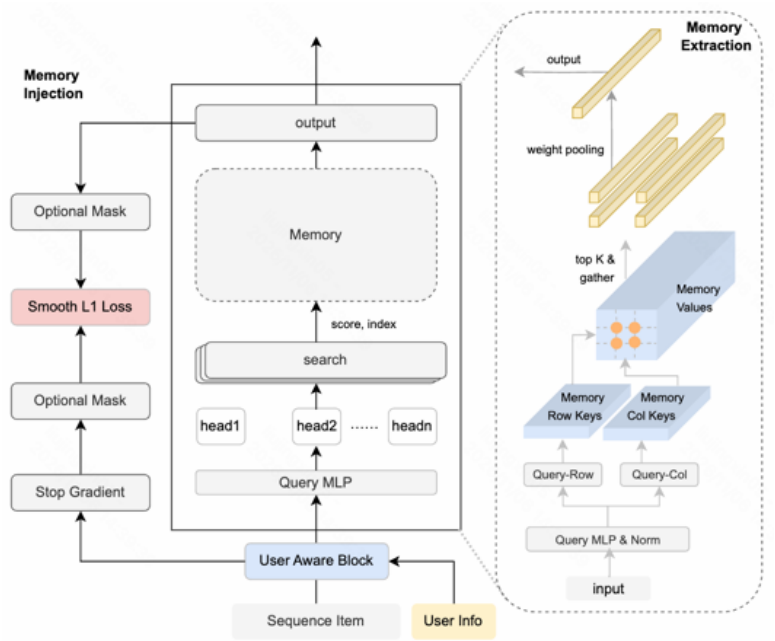


图 20.17: LMN 模型的整体网络结构。

在 Memory Extraction 阶段，LMN 模型首先构建一个包含 n 个记忆槽的大规模 Memory，其中每个记忆槽同时包含一个 Key 和一个 Value，分别组成记忆键矩阵 $\mathbf{M}_K \in \mathbb{R}^{n\times d}$ 和记忆值矩阵 $\mathbf{M}_V \in \mathbb{R}^{n\times d}$，其中 n 表示记忆槽的数量，d 表示记忆向量的维度。对于用户历史行为序列中的物品 x，模型首先将其 Embedding 作为查询，与所有 Memory Key 计算相似度，再通过 Softmax 得到不同记忆槽的激活权重，并对 Memory Value 进行加权求和：

$$\mathbf{M}_O = \text{softmax}\left(\mathbf{x}\mathbf{M}_K^\top\right)\mathbf{M}_V \tag{20.82}$$

其中，$\mathbf{x} \in \mathbb{R}^{1\times d}$ 表示当前历史行为对应的查询向量，$\mathbf{M}_O$ 为读取到的记忆表示。该方式与 Attention 的计算过程类似，通过查询与 Key 之间的相关性，从大量记忆槽中提取有用信息。不过，当 Memory 规模较大时，需要计算查询与全部 n 个 Key 之间的相似度，计算复杂度为 $O(nd)$。随着记忆槽数量不断增加，这种全量检索方式会带来较大的计算开销。

为了提高大规模 Memory 的检索效率，LMN 借鉴 Product Quantization 的思想（关于 PQ 算法的详细介绍参见第18章《向量检索系统》部分），将原始 Memory Key 拆分为行索引和列索引两个部分：

$$\mathbf{M}_K = \mathbf{M}_K^{\text{row}} \times \mathbf{M}_K^{\text{col}} \tag{20.83}$$

其中，$\mathbf{M}_K^{\text{row}} \in \mathbb{R}^{\sqrt{n}\times d}$，$\mathbf{M}_K^{\text{col}} \in \mathbb{R}^{\sqrt{n}\times d}$ 分别表示行索引 Key 和列索引 Key，$\times$ 表示论文中定义的交叉组合操作。这样，原本需要逐一计算的 n 个记忆槽，可以通过两组规模为 $\sqrt{n}$ 的索引进行定位，从而降低相似度计算开销。

不过，推荐场景中的行为语义具有明显的个性化特点，同一个物品在不同用户的兴趣背景下可能具有不同的含义。如果仅根据物品 Embedding 进行记忆检索，可能难以准确区分不同用户对相同物品的兴趣。因此，LMN 模型进一步设计了 User-aware Block，将用户特征引入查询向量的生成过程。LMN 模型首先将当前历史行为 Embedding 与用户侧特征进行拼接，再通过 MLP 生成用户感知的查询表示：

$$\mathbf{M}_Q = \text{MLP}\left(\text{Concat}(\mathbf{x}, \mathbf{u})\right) \tag{20.84}$$

其中，$\mathbf{u}$ 表示用户侧特征，可以包含用户 ID、用户画像等信息。随后，模型通过两个独立的 MLP 分别生成行查询和列查询：

$$\mathbf{M}_Q^{\text{row}} = \text{MLP}_{\text{row}}(\mathbf{M}_Q) \tag{20.85}$$

$$\mathbf{M}_Q^{\text{col}} = \text{MLP}_{\text{col}}(\mathbf{M}_Q) \tag{20.86}$$

在获得两组查询向量后，LMN 模型分别计算它们与行索引 Key 和列索引 Key 之间的相似度：

$$\mathbf{S}_{\text{row}} = \mathbf{M}_Q^{\text{row}}(\mathbf{M}_K^{\text{row}})^\top \tag{20.87}$$

$$\mathbf{S}_{\text{col}} = \mathbf{M}_Q^{\text{col}}(\mathbf{M}_K^{\text{col}})^\top \tag{20.88}$$

随后，通过 Broadcast 操作将两组分数进行组合，得到完整的记忆槽激活分数矩阵：

$$\mathbf{S} = \text{Broadcast}(\mathbf{S}_{\text{row}}, \mathbf{S}_{\text{col}}) \tag{20.89}$$

其中，Broadcast 将行分数和列分数扩展到二维空间，并通过相加得到每个记忆槽对应的激活分数。如果记忆槽由行索引 a 和列索引 b 唯一确定，则其激活分数可以表示为：

$$S_{a,b} = S_{\text{row}}(a) + S_{\text{col}}(b) \tag{20.90}$$

这种二维索引方式只需要分别计算两组规模为 $\sqrt{n}$ 的相似度，其计算复杂度可以由原来的 $O(nd)$ 降低到 $O(\sqrt{n}d)$，从而减少大规模 Memory 的检索开销。在得到完整的激活分数后，LMN 模型选择得分最高的 K 个记忆槽，并通过 Softmax 加权聚合对应的 Memory Value：

$$\mathbf{M}_O = \text{softmax}(\mathbf{S}_{\text{TopK}})\mathbf{M}_{V-\text{TopK}} \tag{20.91}$$

其中，$\mathbf{S}_{\text{TopK}}$ 表示选出的 Top-K 激活分数，$\mathbf{M}_{V-\text{TopK}}$ 表示对应的记忆值。通过这种方式，LMN 模型不需要对整个 Memory 进行完整的 Value 读取，而是只提取与当前用户及历史行为最相关的少量记忆信息。对于需要将行为序列填充到固定长度的场景，模型还可以利用 Mask 排除填充位置对应的无效查询。此外，为了增强模型容量，LMN 模型也支持对查询和 Key 使用 Multi-head 机制，从多个特征子空间进行记忆检索。

在完成 Memory Extraction 之后，LMN 模型还需要解决如何将用户历史交互信息持续写入共享 Memory 的问题。与 MIMN 模型通过显式控制器生成 Add Vector 和 Erase Vector、逐步更新用户级 Memory 的方式不同，LMN 模型并没有采用传统的 Memory Write-back 操作，而是将 Memory Key 和 Memory Value 直接作为可学习参数，通过梯度下降进行更新。模型希望检索得到的记忆表示 $\mathbf{M}_O$ 能够包含当前用户与物品交互所对应的信息，因此引入 Smooth L1 Loss 约束记忆输出与用户感知查询之间的一致性：

$$\mathcal{L}_{\text{Memory}} = \text{SmoothL1}(\mathbf{M}_O, \mathbf{M}_Q) \tag{20.92}$$

通过这一辅助损失，当前用户与物品交互所携带的信息能够通过梯度反向传播，持续更新相关的 Memory Key 和 Memory Value，使共享记忆空间逐渐积累不同用户的历史交互知识。

与传统基于显式读写控制器的记忆网络相比，这种方式将记忆更新转化为常规的参数优化过程，更容易与现有推荐模型的训练流程结合。LMN 模型可以作为一个独立的模块接入现有的 CTR 预估模型，其整体训练目标由原始的 CTR 任务损失和记忆损失共同组成：

$$\mathcal{L}_{\text{LMN}} = \mathcal{L}_{\text{CTR}} + \alpha\mathcal{L}_{\text{Memory}} \tag{20.93}$$

其中，$\mathcal{L}_{\text{CTR}}$ 为原始 CTR 预估任务的损失函数，α 为控制记忆损失权重的超参数。通过这种联合优化的方式，模型既可以利用 CTR 监督信号学习目标预估能力，也能够通过记忆损失持续优化共享 Memory，使其学习到更丰富的用户—物品交互信息。

由于 LMN 模型的 Memory Value 具有很大的参数规模，直接将其全部存放在单台机器或单张 GPU 上会带来较大的存储压力。为了支持大规模 Memory 的线上部署，论文进一步设计了 Memory Parameter Server（MPS），利用多 GPU 的高带宽显存（HBM）对 Memory Value 进行分片存储。图20.18展示了 MPS 的线上部署架构。在模型训练阶段，推荐模型根据 Memory Extraction 得到的激活分数，向 MPS 发起相应的记忆值查询请求；MPS 将不同记忆槽分布在多张 GPU 上，并通过 GPU 之间的 NVLink 实现 All-to-All 数据交换，将所需的 Top-K Memory Value 返回推荐模型。在反向传播时，对应的梯度也通过 All-to-All 操作分发到存储相应记忆槽的 GPU，并完成 Memory 参数更新。在在线推理阶段，由于不再需要反向传播和参数更新，MPS 只需保留记忆值的查询和返回过程，从而支持大规模共享 Memory 的在线检索。

总体而言，LMN 模型通过共享的大规模 Memory 扩展推荐模型的长期信息存储能力，并利用用户感知查询和二维索引机制提高记忆检索效率。Memory Extraction 负责从共享记忆空间中检索与当前用户及物品相关的信

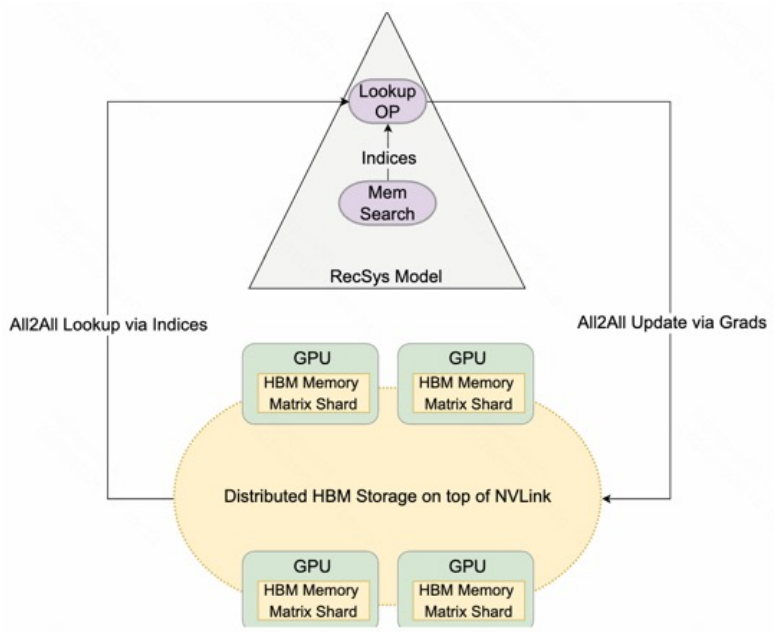


图 20.18: LMN 大规模记忆参数的分布式存储与访问架构。

息，Memory Injection 则通过辅助损失持续将历史交互信息融入记忆参数。MPS 进一步解决了大规模 Memory Value 的分布式存储和在线访问问题。相比起 MIMN 模型为每个用户维护独立记忆并通过 UIC 增量更新的方式，LMN 模型将所有用户的行为信息映射到共享记忆空间中，更强调跨用户的信息复用和模型参数容量扩展。两种方法分别从用户级状态维护和全局共享记忆两个角度探索了基于 Memory 的长序列建模，为工业推荐系统处理大规模历史行为提供了不同的技术方案。

20.4 基于召回检索的用户行为序列建模

前面介绍的基于对比学习、Attention 和 Memory 的用户行为序列建模方法，分别从兴趣表示学习、目标感知的兴趣提取以及历史行为信息压缩等角度提升了模型对用户兴趣的刻画能力。其中，基于对比学习的方法通常通过构造正向行为和隐式负反馈行为，学习具有区分性的用户兴趣表示。例如，可以将用户与候选物品的融合 Embedding 作为查询，分别对用户正向交互行为序列和短时浏览等隐式负反馈序列进行 Target Attention，再通过对比学习约束两类行为对应的兴趣表示；也可以直接对不同类型的行为序列表示进行对比学习。基于 Attention 的方法则主要通过目标感知的注意力机制，从历史行为中提取与当前候选物品相关的兴趣信息。而基于 Memory 的方法则进一步利用固定大小的记忆结构压缩和存储用户长期行为信息。

然而，在工业级推荐系统中，用户历史行为可能达到数千甚至上万条，直接对完整行为序列进行 Attention 计算仍然需要较大的计算开销，而将所有历史行为压缩到固定大小的 Memory 中，又可能损失部分细粒度兴趣信息。**如何在充分利用长期行为信息的同时，减少实时序列建模的计算负担，成为工业推荐系统需要解决的重要问题**。

基于召回检索的用户行为序列建模提供了另一种解决思路。与直接对完整历史行为序列进行建模不同，这类方法首先根据当前候选物品，从超长历史行为中检索出一小部分相关行为，再利用 Attention 等模型对检索结果进行精细化兴趣建模。整个过程通常采用两阶段结构：第一阶段负责快速筛选与目标相关的历史行为，第二阶段负责对筛选后的行为进行更加精细的目标感知建模。这种方式将计算资源集中在更有价值的历史行为上，使原本难以直接处理的超长序列能够应用于工业级 CTR 预估模型。本节将介绍 SIM、TWIN、TWIN-V2、MIRRN、FIM 和 LCN 等代表性方法，重点讨论不同模型在行为检索、序列压缩和目标感知兴趣提取方面的设计。

20.4.1 SIM 模型

SIM（Search-based Interest Model）模型是阿里巴巴提出的基于检索的用户兴趣建模方法，源自于 CIKM 2020 论文《Search-based User Interest Modeling with Lifelong Sequential Behavior Data for Click-Through Rate Prediction》[20]。该模型针对 DIN、DIEN 等 Attention 模型难以直接处理超长用户历史行为的问题，提出了两阶段检索式兴趣建模结构。其核心思路如下：

定义 20.4

SIM 模型先从用户的长期行为序列中检索与当前候选物品相关的行为子集，再对筛选出的行为进行精细化的兴趣建模，从而兼顾长期兴趣信息的利用和在线计算效率。

在传统的 Attention 序列建模方法中，模型通常需要读取用户的完整历史行为序列，并根据当前候选物品计算每个历史行为的相关性。当历史行为长度达到数千甚至上万时，直接进行 Attention 计算不仅会带来较大的计算开销，还会增加线上行为数据的存储和读取压力。SIM 模型认为，对于当前候选物品而言，用户的所有历史行为并不都具有同等价值，真正影响用户当前决策的通常只是其中一部分。因此，与其直接对所有历史行为进行复杂建模，不如先通过相对高效的检索方式筛选出相关行为，再对这些行为进行精细化建模。

SIM 模型的整体网络结构如图20.19所示，采用级联的两阶段检索结构，主要由 General Search Unit（GSU）和 Exact Search Unit（ESU）两个部分组成。GSU 根据当前候选物品，从用户的超长历史行为序列中检索 Top-K 个相关行为，得到长度明显缩短的行为子序列；而 ESU 则以这些相关行为为输入，通过目标感知的 Multi-head Attention 进一步提取用户兴趣。最终，SIM 模型将得到的长期兴趣表示与其他用户、物品及上下文特征结合，输入 MLP 网络完成最终的 CTR 预估。需要注意两个阶段在训练过程中联合优化，而不是完全独立的两个模型。

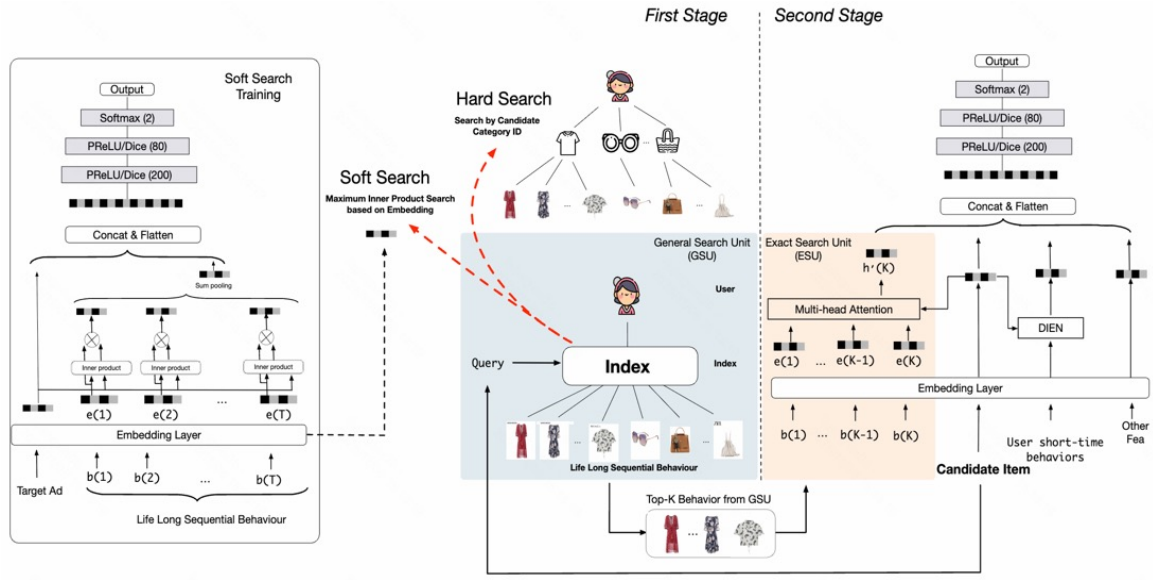


图 20.19: SIM 模型的两阶段检索与兴趣建模结构。

在第一阶段，GSU 负责从原始长期行为序列中筛选与当前候选物品相关的历史行为。假设用户的历史行为序列为：

$$\mathcal{B} = [b_1, b_2, \ldots, b_T] \tag{20.94}$$

其中，T 为历史行为序列长度，b_i 表示第 i 个历史行为，当前候选物品记为 a。GSU 首先计算每个历史行为与候选物品之间的相关性得分 r_i，然后根据得分选取 Top-K 个行为，形成子序列：

$$\mathcal{B}^* = \text{TopK}\left(\mathcal{B}, \{r_i\}_{i=1}^{T}\right) \tag{20.95}$$

然后，SIM 模型根据相关性得分的计算方式，总共设计了两种检索策略：Hard Search 和 Soft Search。其中 Hard Search 是一种非参数化的检索方式，直接利用物品类别信息判断历史行为与候选物品之间的相关性。如果第 i 个

历史行为与当前候选物品属于同一个类别，则认为该行为与候选物品相关，其相关性得分定义为：

$$r_i^{\text{hard}} = \mathbb{I}(C_i = C_a) \tag{20.96}$$

其中，C_i 和 C_a 分别表示历史行为物品和候选物品所属的类别。通过这一方式，模型可以直接筛选出与候选物品属于相同类别的历史行为，无须额外训练复杂的检索网络。Hard Search 的优点是实现简单、检索效率高，尤其适合通过预先构建的索引结构进行线上查询。不过，它主要依赖物品类别判断相关性，无法充分捕获跨类别的潜在兴趣关联。

相比之下，Soft Search 使用可学习的 Embedding 表示计算历史行为与目标物品之间的语义相关性。首先，将历史行为和候选物品映射为低维稠密向量，分别记为 $\mathbf{e}_i$ 和 $\mathbf{e}_a$，然后通过两个线性映射得到相关性得分：

$$r_i^{\text{soft}} = \langle W_b \mathbf{e}_i, W_a \mathbf{e}_a \rangle \tag{20.97}$$

其中，W_b 和 W_a 为可学习的映射矩阵，$\langle \cdot, \cdot \rangle$ 表示向量内积。根据相关性得分，模型选择 Top-K 个历史行为作为后续精细建模的输入。由于用户历史行为可能达到上万条，如果逐一计算所有行为与候选物品之间的相似度，仍然会产生较大的计算开销。因此，SIM 进一步采用 Asymmetric Locality-Sensitive Hashing（ALSH）实现近似最大内积搜索（Maximum Inner Product Search，MIPS），从而在较低的检索开销下，从超长行为序列中快速找到相关行为。

需要注意的是，长期行为和短期行为的数据分布可能存在较大差异，直接使用短期兴趣建模中学习到的参数进行长期行为检索，可能导致检索结果出现偏差。因此，对于 Soft Search，SIM 模型额外设计了基于长期行为数据的辅助 CTR 训练任务，使检索网络能够适应长期行为的分布。首先，模型会根据相关性得分对历史行为 Embedding 进行加权聚合，得到长期行为表示：

$$\mathbf{U}_r = \sum_{i=1}^{T} r_i^{\text{soft}} \mathbf{e}_i \tag{20.98}$$

随后，SIM 模型将 $\mathbf{U}_r$ 与候选物品 Embedding 进行拼接，输入 MLP 进行 CTR 预估，并通过相应的辅助损失训练 Soft Search 的参数。由于完整历史行为序列可能过长，训练时还可以随机采样多个行为子序列，以降低计算开销。在 Soft Search 模式下，GSU 和 ESU 共享部分 Embedding 参数，并通过联合训练不断更新检索所使用的表示，使检索结果能够随模型参数的变化持续优化。

经过第一阶段的筛选，GSU 得到与当前候选物品相关的行为子序列 $\mathcal{B}^*$。第二阶段的 ESU 则负责对这些行为进行更加精细的目标感知兴趣建模。由于检索出的行为可能分布在用户较长的历史时间范围内，不同行为距离当前候选物品的时间间隔也可能不同。因此，SIM 模型也在 ESU 模块中额外引入了时间间隔的特征。对于检索得到的第 j 个行为，记其与当前候选物品之间的时间间隔为 Δ_j，通过 Embedding 层得到对应的时间表示 $\mathbf{e}_j^t$，并与行为本身的 Embedding $\mathbf{e}_j^*$ 进行拼接：

$$\mathbf{z}_j = \text{Concat}(\mathbf{e}_j^*, \mathbf{e}_j^t) \tag{20.99}$$

这样 ESU 模块不仅能够利用历史行为的语义信息，还可以区分不同行为距离当前推荐目标的时间远近。

在此基础上，ESU 模块使用 Multi-head Attention 对检索后的行为进行进一步建模。对于第 h 个 Attention Head，首先根据当前候选物品和历史行为的表示计算相关性得分：

$$s_j^{(h)} = \left\langle \mathbf{W}_b^{(h)} \mathbf{z}_j, \mathbf{W}_a^{(h)} \mathbf{e}_a \right\rangle \tag{20.100}$$

然后通过 Softmax 归一化得到各个历史行为的注意力权重：

$$\alpha_j^{(h)} = \frac{\exp(s_j^{(h)})}{\sum_{l=1}^{K} \exp(s_l^{(h)})} \tag{20.101}$$

每个 Attention Head 根据注意力权重对检索后的行为表示进行加权聚合：

$$\text{head}_h = \sum_{j=1}^{K} \alpha_j^{(h)} \mathbf{z}_j \tag{20.102}$$

最终，将多个 Attention Head 的输出进行拼接，得到用户长期兴趣表示：

$$\mathbf{U}_{lt} = \text{Concat}(\text{head}_1, \ldots, \text{head}_q) \tag{20.103}$$

其中，q 为 Attention Head 的数量。通过上述设计，ESU 能够在经过初步筛选的历史行为中进一步提取与当前候选物品相关的多维兴趣信息，并结合时间间隔特征刻画不同行为对当前推荐目标的影响。

在训练过程中，GSU 与 ESU 通过联合目标进行优化：

$$\mathcal{L} = \alpha \mathcal{L}_{\text{GSU}} + \beta \mathcal{L}_{\text{ESU}} \tag{20.104}$$

其中，$\mathcal{L}_{\text{GSU}}$ 和 $\mathcal{L}_{\text{ESU}}$ 分别表示两个阶段对应的 CTR 预估损失，α 和 β 为损失权重。对于 Soft Search，两个阶段均通过可学习参数参与联合训练。而对于 Hard Search，由于检索过程是基于类别匹配的非参数操作，不需要对 GSU 进行梯度优化，因此可以只通过 ESU 的预测目标来训练后续的网络。

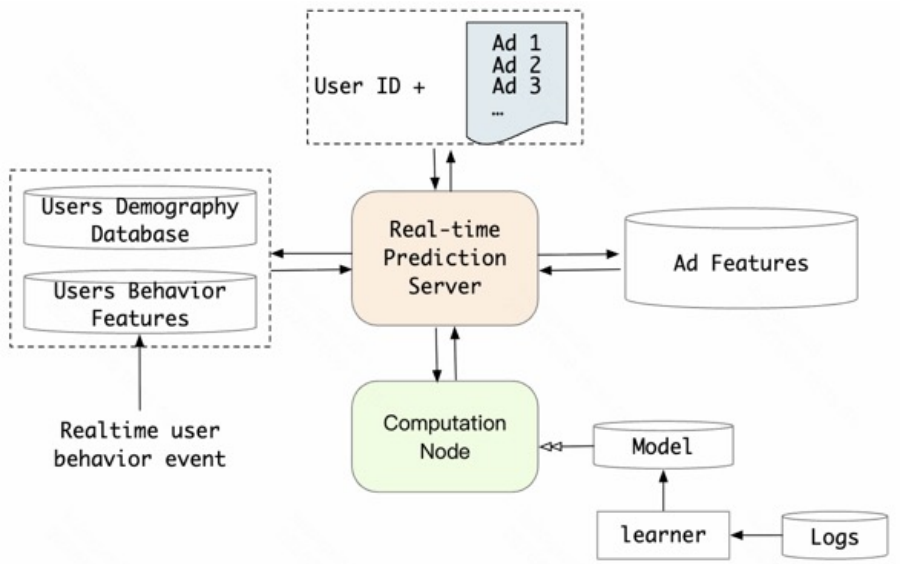


图 20.20: 传统 CTR 实时预估系统的服务流程。

在完成模型设计之后，SIM 模型还针对工业线上服务进行了专门优化。图20.20展示了论文中传统实时预估系统的基本结构，主要包括计算节点和预测服务。历史行为数据通常存储在额外的分布式存储系统中，当推荐请求到达时，计算节点需要读取用户历史行为，并将相关特征发送给预测服务完成模型推理。随着用户行为序列长度增加，历史行为数据的存储、传输和实时计算开销都会相应增长。当序列长度达到数千甚至上万时，传统架构很难同时满足线上服务的延迟和吞吐量要求。

为解决上述模型推理方面的问题，SIM 模型在实际部署中选择了 Hard Search 作为 GSU 模块的主要实现方式，并针对用户历史行为构建了 User Behavior Tree（UBT）索引结构。论文的离线实验发现，Soft Search 检索得到的 Top-K 行为与 Hard Search 的结果具有较高相似性，大部分 Soft Search 选出的行为也属于与候选物品相同的类别。虽然 Soft Search 在离线实验中取得了略好的效果，但考虑到线上检索效率和资源消耗，最终选择 Hard Search 进行部署。

UBT 采用了 Key-Key-Value 的多级索引结构：第一级 Key 为用户 ID，第二级 Key 为物品类别 ID，Value 则存储该用户在对应类别下发生过的具体物品行为。由于用户历史行为天然带有物品类别信息，因此可以提前根据用户 ID 和类别 ID 建立索引。在实时预测过程中，以当前候选物品的类别作为检索条件，就能够直接从 UBT 中获取该用户对应类别下的历史行为，而不需要遍历完整的长期行为序列。

图20.21展示了引入 SIM 模型后的线上预估架构。与传统系统相比，新架构增加了基于 UBT 的 Hard Search 模块，用于在实时预测阶段快速筛选候选物品相关的历史行为。UBT 索引可以提前通过离线任务构建，线上只需要根据用户 ID 和目标物品类别查询对应的行为列表，因此大部分索引构建成本不会出现在实时请求链路中。经过 GSU 检索后，原本可能超过一万条的用户历史行为能够缩减到数百条，再交由 ESU 进行精细化兴趣建模。其他用户及上下文特征的计算也可以与检索过程并行执行，从而进一步减少整体请求延迟。论文中 UBT 的分布式索引规模达到约 22 TB，体现了这一检索方案在大规模工业环境中的工程实现能力。

总的来说，SIM 模型的主要贡献在于提出了面向超长用户行为序列的两阶段检索建模范式。GSU 通过 Hard

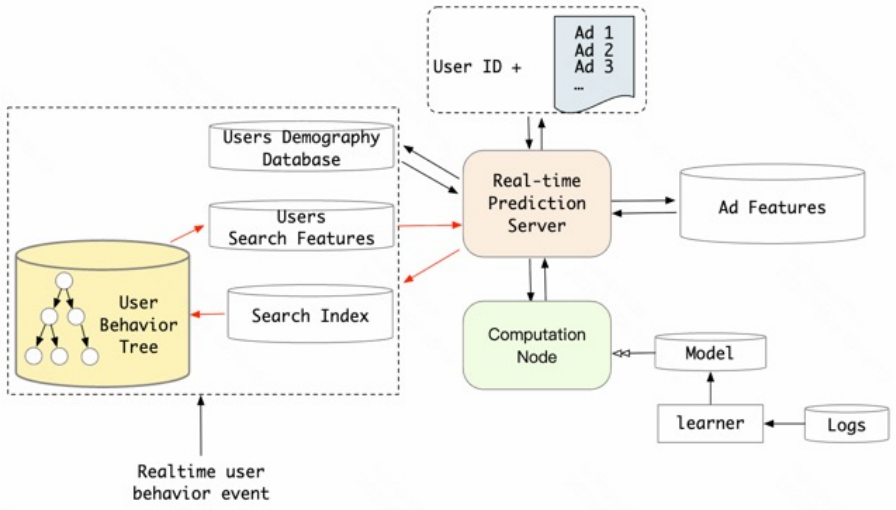


图 20.21: 基于 SIM 检索机制的线上 CTR 预估架构。

Search 或 Soft Search 从完整历史行为中快速筛选与目标物品相关的行为，降低了后续兴趣建模的计算规模。ESU 则利用时间间隔编码和 Multi-head Attention，对筛选后的行为进行更加精细的目标感知建模。在工业落地方面，SIM 模型进一步通过 UBT 将类别检索转化为高效的索引查询，使超长历史行为能够在严格的在线服务延迟约束下得到利用。与直接对完整序列进行 Attention 建模的方法相比，SIM 模型将复杂的兴趣建模集中在少量检索结果上，为后续 TWIN 等检索式序列建模方法提供了重要的技术基础。

20.4.2 TWIN 模型

TWIN（TWo-stage Interest Network）模型是快手科技提出的面向用户终身行为建模的 CTR 预估模型，源自于 KDD 2023 会议论文《TWIN: TWo-stage Interest Network for Lifelong User Behavior Modeling in CTR Prediction at Kuaishou》[3]。该模型与 SIM 模型一样，采用两阶段检索结构，从用户超长历史行为序列中筛选与当前候选物品相关的行为，再进行精细化的目标感知兴趣建模。不过，与 SIM 模型通过相对轻量的 GSU 完成初步筛选不同，TWIN 模型进一步强调两个阶段之间的相关性一致性，提出 Consistency-Preserved General Search Unit（CP-GSU），使粗粒度检索和精细化兴趣建模采用相同的目标行为相关性度量，从而减少两阶段检索标准不一致所带来的信息偏差。

在 SIM 模型中，GSU 可以通过物品类别匹配等较为简单的方式快速筛选相关行为，而 ESU 则利用 Multi-head Attention 对筛选后的行为进行精细化建模。这种设计虽然能够有效降低计算开销，但初步检索阶段使用的相关性度量与最终兴趣建模阶段并不完全一致，可能导致部分对最终预测有价值的历史行为在第一阶段就被过滤掉了。

TWIN 模型希望能够直接将精细化建模阶段使用的 Multi-Head Target Attention（MHTA）扩展到超长行为序列上，使得两个阶段采用相同的相关性计算方式。然而，传统 MHTA 需要对所有历史行为进行较为复杂的线性投影，当历史行为长度达到上万时，其计算开销难以满足实时预估的要求。因此，**如何在保持检索相关性度量一致的同时，大幅降低 MHTA 在超长序列上的计算开销，是 TWIN 模型需要解决的核心问题**。

TWIN 模型在快手 CTR 预估模型中的整体结构如图20.22所示。整个模型除了常规的用户、物品和上下文特征外，还包含三个主要的特征建模模块：TWIN 长期行为建模模块、短期行为建模模块以及其他特征建模模块。其中，TWIN 长期行为建模模块负责从数万条长期历史行为中检索与当前目标相关的行为，并通过两阶段建模提取长期兴趣；短期行为建模模块则直接利用最近的 50 条行为提取用户近期兴趣，作为长期兴趣建模的补充；其他特征建模模块负责处理用户画像、视频属性以及上下文等信息。三个模块的输出最终通过上层神经网络进行融合，共同完成 CTR 预估任务。

TWIN 模型的核心在于如何高效地将 MHTA 扩展到超长历史行为序列上。传统 MHTA 主要通过目标物品与历史行为之间的相关性计算注意力权重，再对行为表示进行加权聚合。在处理较短的行为序列时，这种方式能够

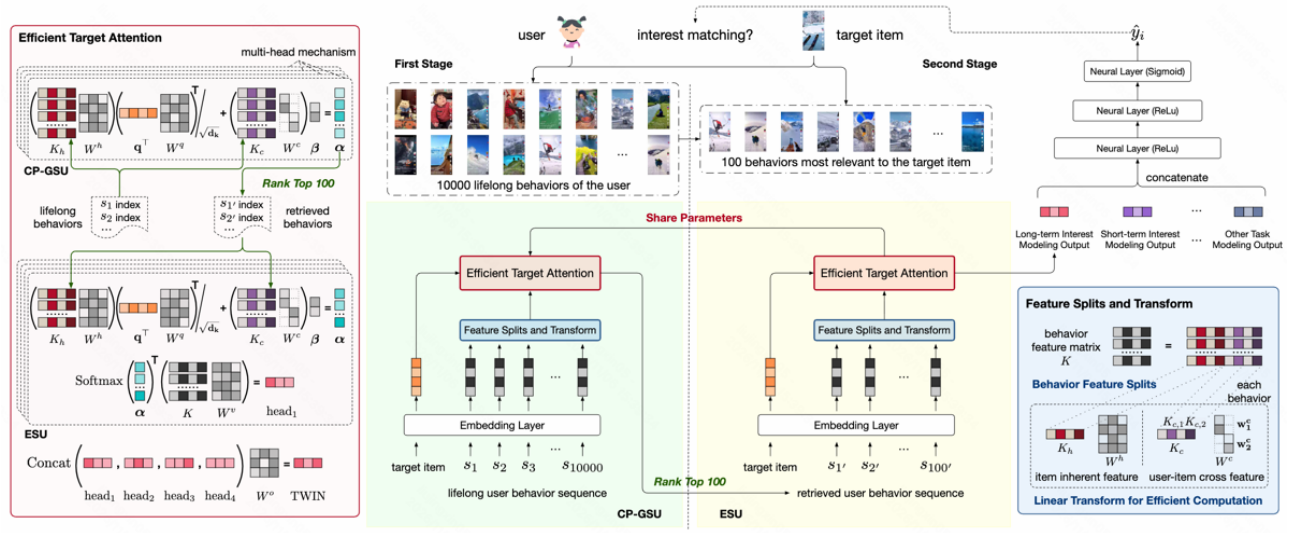


图 20.22: TWIN 在快手 CTR 预估模型中的整体架构。

有效提取候选物品感知的用户兴趣。但当历史行为达到数万条时，对所有行为特征进行完整的线性投影会成为主要的计算瓶颈。针对这一问题，TWIN 模型首先将历史行为特征拆分为两类：物品固有特征（Inherent Features）和用户—物品交叉特征（User-item Cross Features）。

假设长度为 L 的历史行为序列对应的特征矩阵为 $\mathbf{K}$，可以将其拆分为：

$$\mathbf{K} = [\mathbf{K}_h, \mathbf{K}_c] \in \mathbb{R}^{L\times(H+C)} \tag{20.105}$$

其中，$\mathbf{K}_h \in \mathbb{R}^{L\times H}$ 表示物品固有特征，例如视频 ID、作者 ID、主题和视频时长等，这些特征主要由物品自身属性决定，不依赖于具体用户；$\mathbf{K}_c \in \mathbb{R}^{L\times C}$ 表示用户与物品之间的交叉特征，例如用户点击视频的时间、播放时长和浏览页面位置等，这些特征反映了用户与具体物品之间的交互情况。

对于物品固有特征，由于同一个物品在不同用户的历史行为序列中具有相同的固有属性，其对应的线性投影结果可以在不同用户和请求之间复用。因此，TWIN 模型将这部分特征的线性投影提前计算，并通过缓存机制进行维护。对于用户—物品交叉特征，由于其具有明显的个性化特点，无法像物品固有特征一样直接共享缓存。因此，TWIN 模型进一步简化这部分特征的线性投影过程。

假设包含 J 个用户—物品交叉特征，每个特征的 Embedding 维度为 8，则交叉特征矩阵可以拆分为：

$$\mathbf{K}_c = [\mathbf{K}_{c,1}, \ldots, \mathbf{K}_{c,J}] \tag{20.106}$$

其中，$\mathbf{K}_{c,j} \in \mathbb{R}^{L\times 8}$ 表示第 j 个交叉特征对应的矩阵。传统的线性投影需要使用完整的权重矩阵将所有交叉特征映射到目标维度，而 TWIN 模型将其简化为每个交叉特征单独使用一个向量进行投影：

$$\mathbf{K}_c\mathbf{W}^c = [\mathbf{K}_{c,1}\mathbf{w}_1^c, \ldots, \mathbf{K}_{c,J}\mathbf{w}_J^c] \tag{20.107}$$

其中，$\mathbf{w}_j^c \in \mathbb{R}^8$ 为第 j 个交叉特征对应的投影参数。这样，每个交叉特征最终被压缩为一个标量，整体投影维度从原来的 $L\times C$ 降低为 $L\times J$。从矩阵结构来看，这相当于对原始投影矩阵施加了分块对角约束，从而显著减少交叉特征投影的计算量。

在完成特征拆分与投影简化后，TWIN 模型进一步构建统一的目标行为相关性计算方式，使其同时适用于 CP-GSU 和 ESU。假设当前候选物品的固有特征表示为 $\mathbf{q} \in \mathbb{R}^H$，经过线性映射后，与历史行为的固有特征投影计算 Scaled Dot-Product 相关性。同时，将用户—物品交叉特征的投影结果作为额外的偏置项，最终得到每个历史行为与当前候选物品之间的相关性得分：

$$\boldsymbol{\alpha} = \frac{(\mathbf{K}_h\mathbf{W}^h)(\mathbf{q}^\top\mathbf{W}^q)^\top}{\sqrt{d_k}} + (\mathbf{K}_c\mathbf{W}^c)\boldsymbol{\beta} \tag{20.108}$$

其中，$\mathbf{W}^h$ 和 $\mathbf{W}^q$ 分别为行为 Key 和目标 Query 对应的投影矩阵，d_k 为投影后的特征维度，$\boldsymbol{\beta}$ 为交叉特征偏置项的可学习权重。该公式由两部分组成：第一部分通过目标物品与历史行为的固有特征计算语义相关性，第二

部分则利用用户与物品之间的交叉特征对相关性得分进行调整。通过这种方式，模型既保留了基于目标与行为固有特征的匹配能力，也能够利用具体用户的交互信息实现个性化修正。

在得到相关性得分 $\boldsymbol{\alpha}$ 后，CP-GSU 直接从数万条长期历史行为中选择得分最高的 100 条，形成后续精细建模所需要的行为子序列。与 SIM 模型的 Hard Search 不同，CP-GSU 并不依赖简单的类别匹配规则，而是采用与 ESU 相同的目标行为相关性度量进行筛选。与此同时，由于物品固有特征的投影已经提前计算并缓存，交叉特征的投影也被简化，因此即使面对上万条历史行为，仍然能够高效地计算相关性得分。这也是 TWIN 模型能够在保持两阶段检索标准一致的情况下，将目标感知检索扩展到超长行为序列的关键。

在第二阶段，ESU 使用与 CP-GSU 相同的相关性计算方式，对第一阶段筛选出的 100 条历史行为进行精细化兴趣建模。ESU 首先根据相关性得分计算归一化注意力权重，再对历史行为的 Value 表示进行加权聚合：

$$\text{Attention} = \text{softmax}(\boldsymbol{\alpha})^\top \mathbf{K}\mathbf{W}^v \tag{20.109}$$

其中，$\mathbf{W}^v$ 为 Value 对应的投影矩阵。由于此时输入序列已经被缩短到 100 条，完整的 Value 投影和注意力计算可以在实时预测过程中高效执行。为了从不同表示子空间中提取用户兴趣，TWIN 模型进一步采用 4 个 Attention Head 分别计算目标感知兴趣，并将各个 Head 的输出进行拼接和线性映射：

$$\text{TWIN} = \text{Concat}(\text{head}_1, \ldots, \text{head}_4)\mathbf{W}^o \tag{20.110}$$

其中，第 a 个 Attention Head 使用对应的 Query、Key 和 Value 投影参数计算注意力结果，$\mathbf{W}^o$ 为多头输出的线性映射矩阵。

需要强调的是，TWIN 模型的 CP-GSU 和 ESU 并不是采用两套不同的相关性计算规则，而是使用相同的 MHTA 结构和参数进行目标行为匹配。两者的区别主要在于处理的历史行为规模以及计算方式：CP-GSU 面向数万条行为，通过预计算和简化投影高效获得相关性分数，并执行 Top-K 检索；ESU 则对筛选后的少量行为进行完整的目标感知兴趣聚合。**通过这种一致性设计，CP-GSU 所筛选出的行为能够更好地对应 ESU 实际关注的相关行为，减少由于两阶段相关性标准不一致而产生的信息损失**。这也是 TWIN 模型相比起 SIM 模型最重要的改进。

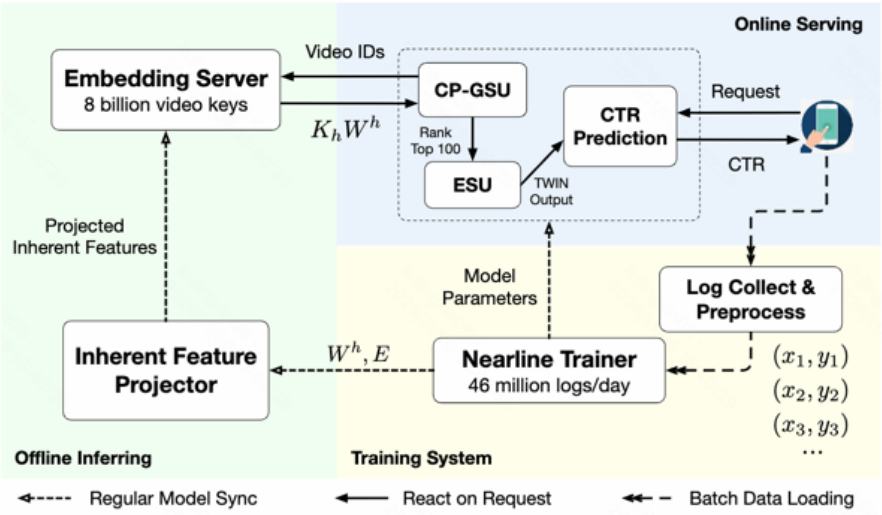


图 20.23: TWIN 模型在线服务中的特征预计算与缓存架构。

在完成模型设计后，TWIN 模型进一步通过离线预计算和在线缓存机制解决大规模行为检索中的计算瓶颈。图20.23展示了 TWIN 模型在快手线上 CTR 预估系统中的部署架构。整个系统主要包含三个部分：近线训练系统、离线推理系统和在线服务系统。近线训练系统持续消费用户与视频之间的交互日志，并联合训练完整 CTR 模型及 TWIN 模块；模型参数通过消息队列定期同步到离线推理和在线服务系统。论文中的近线训练系统能够在交互日志产生后 8 分钟以内完成数据处理并用于模型训练，模型参数则大约每 5 分钟同步一次。

其中，离线推理系统的核心任务是提前计算和缓存物品固有特征的线性投影结果。由于同一个视频的固有特征在不同用户行为序列中可以共享，系统可以周期性地获取最新的物品 Embedding 和 TWIN 投影参数，批量

计算候选视频对应的投影结果，并将其存储到 Embedding Server 中，形成以视频 ID 为 Key、投影结果为 Value 的查询服务。为了控制缓存规模，论文对候选视频池进行了长尾裁剪，将索引规模控制在约 80 亿个视频，并通过每 15 分钟刷新一次缓存来减小参数更新带来的表示偏差。该缓存能够覆盖约 97% 的线上请求，在计算效率与覆盖率之间取得平衡。

当在线服务收到推荐请求时，首先向 Embedding Server 查询候选视频对应的物品固有特征投影结果，再结合当前用户行为序列中的交叉特征，在实时请求链路中计算目标行为相关性得分。随后，系统根据相关性得分选出 Top-100 历史行为，并将其输入 ESU 进行完整的多头目标感知兴趣建模。由于 ESU 只处理少量历史行为，因此相关的 Value 投影和后续兴趣聚合可以使用最新同步的模型参数在线计算。通过将数万条历史行为上的主要线性投影操作转移到离线预计算阶段，TWIN 模型在实际部署中将相关计算瓶颈降低了约 99.3%。

总体而言，TWIN 模型的核心贡献在于通过特征拆分、投影简化和离线预计算，使 MHTA 能够以相同的相关性度量同时服务于粗粒度检索和精细化兴趣建模。CP-GSU 负责从数万条历史行为中快速筛选相关行为，ESU 则进一步利用多头目标感知注意力提取用户兴趣，从而在检索效率和兴趣建模准确性之间取得平衡。相比起 SIM 模型采用 Hard Search 或 Soft Search 实现初步筛选的方式，TWIN 模型更强调两阶段相关性度量的一致性，并通过缓存和预计算解决大规模 MHTA 的工程计算瓶颈。TWIN 模型在快手主站推荐排序系统中实现了全量部署，服务约 3.46 亿日活用户，峰值处理请求达到每秒 3000 万个视频，体现了该方法在大规模工业推荐场景中的应用价值。

20.4.3 TWIN-V2 模型

TWIN-V2 模型是快手科技在 TWIN 模型的基础上进一步提出的超长用户行为序列建模方法，源自于 CIKM 2024 会议论文《TWIN V2: Scaling Ultra-Long User Behavior Sequence Modeling for Enhanced CTR Prediction at Kuaishou》[21]。TWIN 模型通过两阶段目标感知检索，从数万条历史行为中筛选与当前候选物品相关的行为，并进行精细化兴趣建模。不过，在短视频推荐场景中，用户的历史行为会随着使用时间不断累积，只依靠最近几个月的行为仍然可能遗漏长期兴趣信息。随着用户生命周期行为序列进一步增长，即使采用 TWIN 模型的高效检索机制，直接处理完整的生命周期行为也会带来较大的存储和计算压力。因此，TWIN-V2 模型在 TWIN 模型的基础上引入层次聚类机制，先对用户整个生命周期内的历史行为进行离线压缩，再利用两阶段检索从压缩后的行为中提取长期兴趣。

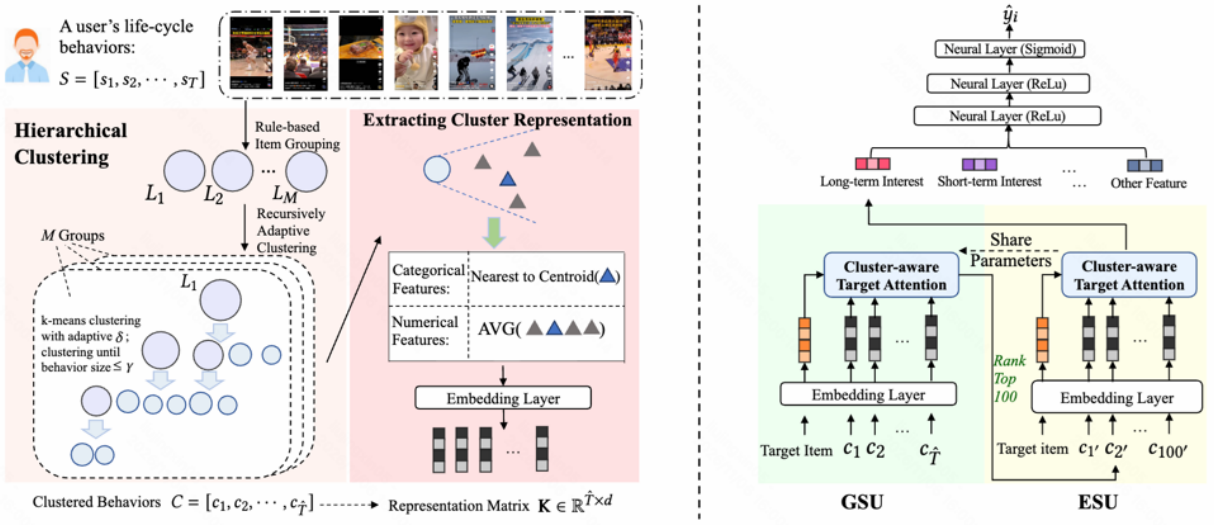


图 20.24: TWIN-V2 模型的生命周期行为压缩与两阶段兴趣建模流程。

TWIN-V2 模型的整体流程如图20.24所示，主要分为离线处理和在线推理两个阶段。离线阶段首先对用户生命周期内的历史行为进行层次聚类，将相似的历史物品聚合为多个 Cluster，并为每个 Cluster 提取一个虚拟物品表示，从而将原始超长行为序列压缩为较短的聚类序列。在线阶段则以这些 Cluster 为输入，继续采用 TWIN 模

型的两阶段检索结构：GSU 从压缩后的行为中筛选与目标物品相关的 Top-100 个 Cluster，ESU 再利用目标感知 Attention 对这些 Cluster 进行精细化兴趣建模。与 TWIN 模型类似，两个阶段均采用高效的目标感知注意力计算机制，同时进一步考虑不同 Cluster 所包含的原始行为数量，以提高长期兴趣提取的准确性。

在生命周期行为压缩阶段，TWIN-V2 模型首先对用户历史行为进行分组，然后通过递归聚类将相似物品组织到不同的 Cluster 中。假设用户的生命周期行为序列为：

$$S = [s_1, s_2, \ldots, s_T] \tag{20.111}$$

其中，T 表示用户历史行为的总数量，s_j 表示第 j 个交互物品。模型首先根据每个物品的播放完成率（Playing Completion Ratio）对历史行为进行分组。对于第 j 个行为，其播放完成率（完播率）定义为：

$$p_j = \frac{\text{播放时长}(s_j)}{\text{视频时长}(s_j)} \tag{20.112}$$

模型将历史行为划分为 M 个不同的分组，使同一组内的物品具有相近的播放完成率。在实际应用中，M 的取值可以设置为 5。引入这一分组步骤，是因为播放完成率在一定程度上能够反映用户对视频内容的兴趣程度。如果直接对所有行为进行 K-means 聚类，由于 K-means 主要依据物品 Embedding 之间的距离进行划分，可能将播放反馈差异较大的行为分配到同一个 Cluster 中。因此，先根据播放完成率进行分组，可以使后续聚类结果在用户反馈强度上具有更好的同质性。

在完成初始分组后，TWIN-V2 模型进一步对每个分组中的行为进行递归 K-means 聚类。为了适应不同用户的行为规模，TWIN-V2 模型没有为所有用户设置固定的聚类数量，而是根据当前待聚类集合中的物品数量动态确定聚类数。假设当前待处理的物品集合为 V，则本次聚类的簇数量设置为：

$$\delta = \lfloor |V|^{0.3} \rfloor \tag{20.113}$$

随后，模型根据推荐模型学习得到的物品 Embedding，对集合 V 执行 K-means，将其划分为 δ 个子集合，并对每个子集合递归执行相同的操作。当某个集合中的物品数量低于设定的最大聚类规模 γ 时，则停止继续划分，并将该集合视为最终的 Cluster。通过这种递归聚类方式，不同用户可以根据自身历史行为的规模形成不同数量的 Cluster，而不需要采用统一的聚类迭代次数。

经过上述处理，原始生命周期行为序列被压缩为较短的 Cluster 序列：

$$C = h_{\text{comp}}(S) = [c_1, c_2, \ldots, c_{\hat{T}}] \tag{20.114}$$

其中，c_i 表示由多个相似历史物品组成的第 i 个 Cluster，$\hat{T}$ 表示聚类后的序列长度，通常远小于原始行为长度 T。在实际应用中，最大聚类规模 γ 可以设置为 20，最终平均每个 Cluster 包含约 10 个历史行为，生命周期行为序列可以压缩至原始长度的约 10%。

在完成聚类之后，还需要将每个 Cluster 转换为模型能够直接使用的特征表示。由于一个 Cluster 中包含多个历史物品，如果直接保留所有物品特征，仍然会带来较大的存储和计算开销。因此，TWIN-V2 模型为每个 Cluster 构造一个虚拟物品（Virtual Item），使用固定长度的特征表示整个 Cluster。

TWIN-V2 模型将物品特征分为类别特征和数值特征两类，并分别采用不同的聚合方式。对于数值特征，例如视频时长和用户播放时长，模型直接计算 Cluster 内所有物品对应特征的平均值：

$$\mathbf{c}_i^{\text{num}} = \frac{1}{|c_i|} \sum_{v \in c_i} \mathbf{x}_v^{\text{num}} \tag{20.115}$$

其中，$\mathbf{x}_v^{\text{num}}$ 表示物品 v 的数值特征。对于类别特征，例如视频 ID 和作者 ID，由于不同物品的类别特征通常无法直接进行数值平均，模型选择距离 Cluster 中心最近的物品作为代表。模型首先根据 Cluster 中所有物品的 Embedding 计算聚类中心，再选取距离中心最近的物品：

$$v_i^* = \arg\min_{v \in c_i} \|\mathbf{k}_v - \mathbf{k}_{\text{centroid}}\|_2^2 \tag{20.116}$$

其中，$\mathbf{k}_v$ 和 $\mathbf{k}_{\text{centroid}}$ 分别表示物品 v 和聚类中心对应的 Embedding。最终，模型使用该代表物品的类别特征作为整个 Cluster 的类别特征表示：

$$\mathbf{c}_i^{\text{cat}} = \mathbf{x}_{v_i^*}^{\text{cat}} \tag{20.117}$$

通过将两类特征组合起来，可以得到每个 Cluster 对应的虚拟物品表示：

$$\mathbf{c}_i = \text{Concat}(\mathbf{c}_i^{\text{cat}}, \mathbf{c}_i^{\text{num}}) \tag{20.118}$$

这些虚拟物品经过 Embedding 层转换后，得到对应的 Cluster 表示 $\mathbf{k}_i$，最终形成压缩后的行为特征矩阵。值得注意的是，聚类过程中使用的物品 Embedding 来自推荐模型本身，因此聚类结果不仅考虑了物品内容或类别上的相似性，也能够利用模型从用户交互数据中学习到的协同过滤信息。

在得到 Cluster 表示之后，TWIN-V2 模型继续沿用 TWIN 模型的两阶段检索思路。但 TWIN-V2 模型进一步改进了目标感知注意力，提出了 Cluster-aware Target Attention，使其能够考虑不同 Cluster 所包含的原始行为数量。模型首先将 Cluster 特征拆分为物品固有特征和用户—物品交叉特征，并采用 TWIN 模型中的高效特征投影方法计算 Cluster 与目标物品之间的相关性：

$$\boldsymbol{\alpha} = \frac{(\mathbf{K}_h \mathbf{W}^h)(\mathbf{q}^\top \mathbf{W}^q)^\top}{\sqrt{d_k}} + (\mathbf{K}_c \mathbf{W}^c)\boldsymbol{\beta} \tag{20.119}$$

其中，$\mathbf{K}_h$ 和 $\mathbf{K}_c$ 分别表示聚类行为的固有特征和交叉特征，$\mathbf{q}$ 表示当前目标物品的固有特征，$\mathbf{W}^h$、$\mathbf{W}^q$ 和 $\mathbf{W}^c$ 为可学习的投影参数，d_k 为投影维度，$\boldsymbol{\beta}$ 为交叉特征对应的权重，$\boldsymbol{\alpha}$ 表示各个 Cluster 与目标物品之间的原始相关性得分。

不过，仅根据原始相关性得分进行排序，还没有充分考虑 Cluster 内部所包含的历史行为数量。例如，如果两个 Cluster 与目标物品具有相同的相关性得分，但其中一个 Cluster 包含 10 个历史行为，另一个只包含 2 个历史行为，那么前者通常包含更丰富的用户兴趣证据。为了反映这种差异，TWIN-V2 模型根据 Cluster 的规模对相关性得分进行修正：

$$\alpha_i' = \alpha_i + \ln n_i \tag{20.120}$$

其中，$n_i = |c_i|$ 表示第 i 个 Cluster 所包含的原始历史行为数量，α_i' 为调整后的相关性得分。经过这一调整，Cluster 的注意力权重可以表示为：

$$\text{Softmax}(\alpha_i') = \frac{n_i \exp(\alpha_i)}{\sum_{j=1}^{\hat{T}} n_j \exp(\alpha_j)} \tag{20.121}$$

从上述公式可以看出，Cluster-aware Target Attention 相当于在原始相关性得分的基础上引入 Cluster Size 权重，使包含更多历史行为的 Cluster 在其他条件相近时获得更大的注意力权重。这样，模型不仅考虑了 Cluster 与目标物品之间的语义相关性，还能够利用 Cluster 的规模信息反映用户在相应兴趣上的行为积累程度。

因此，在第一阶段，GSU 根据调整后的相关性得分，从压缩后的生命周期行为中选择 Top-100 个 Cluster，将其作为后续精细化建模的输入。相比起 TWIN 模型直接从原始历史行为中选择 100 个物品，TWIN-V2 模型选择的是 100 个 Cluster，每个 Cluster 又包含多个原始历史行为，因此能够在相同的检索数量下覆盖更加丰富的用户长期兴趣。在第二阶段，ESU 使用与 GSU 相同的 Cluster-aware Target Attention 对筛选出的 Cluster 进行进一步聚合，得到用户长期兴趣表示：

$$\text{Attention} = \text{Softmax}(\boldsymbol{\alpha}')^\top \mathbf{K}\mathbf{W}^v \tag{20.122}$$

其中，$\mathbf{K}$ 为 Cluster 表示矩阵，$\mathbf{W}^v$ 为 Value 投影矩阵。与 GSU 不同，ESU 需要对筛选出的 Cluster 进行更加精细的兴趣聚合，最终通过 4 个 Attention Head 从不同表示子空间提取长期兴趣：

$$\mathbf{U}_{lt} = \text{Concat}(\text{head}_1, \ldots, \text{head}_4)\mathbf{W}^o \tag{20.123}$$

其中，$\mathbf{U}_{lt}$ 表示最终得到的用户长期兴趣表示，$\mathbf{W}^o$ 为多头输出的线性映射矩阵。通过上述过程，TWIN-V2 模型将原本需要直接处理的超长历史行为转换为数量较少的 Cluster，再通过两阶段目标感知检索提取用户长期兴趣。

在实际部署方面，TWIN-V2 模型将生命周期行为的处理划分为离线和在线两个阶段，整体架构如图20.25所示。离线阶段主要负责历史行为的周期性压缩和 Cluster 特征提取。考虑到快手用户规模较大，不可能在每次推荐请求到达时重新对用户完整生命周期行为进行聚类，因此 TWIN-V2 模型采用周期性离线处理的方式，每两周对用户生命周期行为执行一次完整的层次聚类，并将得到的 Cluster 转换为虚拟物品特征。如果将最大聚类规模 γ 设置为 20，平均每个 Cluster 包含约 10 个行为，最终可以将历史行为压缩至原始长度的约 10%，相关存储开销也能减少约 90%。

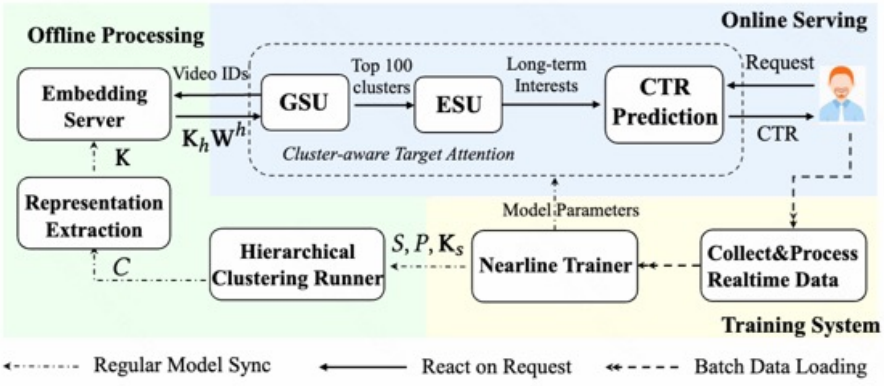


图 20.25: TWIN-V2 模型在快手推荐系统中的离线处理与在线推理架构。

TWIN-V2 模型的在线阶段则沿用了 TWIN 模型中的预计算和缓存机制。离线聚类结果被输入在线兴趣建模流程，首先通过 GSU 计算 Cluster 与目标物品之间的相关性，并筛选 Top-100 个 Cluster。随后，ESU 对这些 Cluster 进行目标感知兴趣聚合，生成用户长期兴趣表示，供后续 CTR 模型使用。与此同时，系统继续采用 TWIN 模型中的 Inherent Feature Projector，对物品固有特征的投影结果进行预计算，并通过 Embedding Server 提供缓存查询服务。为了保证缓存表示与最新模型参数之间的一致性，系统每 15 分钟根据最新 CTR 模型参数刷新相应的预计算结果。

需要注意的是，TWIN-V2 模型并不是对原有 TWIN 模型的完全替代。在线上实际部署中，快手采用两者协同的方式：TWIN 模型主要建模最近几个月的用户历史行为，而 TWIN-V2 模型则负责覆盖用户整个生命周期内的行为信息。两者分别从近期兴趣和长期兴趣两个时间范围提供互补的用户兴趣表示，从而使推荐模型能够同时利用用户近期的内容偏好和长期积累的兴趣信息。

总体而言，TWIN-V2 模型在 TWIN 模型两阶段目标感知检索的基础上，引入了面向生命周期行为的层次聚类和 Cluster-aware Target Attention。前者通过行为分组和递归聚类，将大量相似历史行为压缩为具有代表性的虚拟物品；后者通过引入 Cluster Size 修正相关性得分，使模型在检索和聚合过程中能够考虑不同 Cluster 所包含的历史行为规模。结合离线聚类、特征预计算和在线两阶段检索，TWIN-V2 模型将用户行为建模范围从最近几个月进一步扩展到整个生命周期，为工业推荐系统中的超长序列建模提供了一种有效的解决方案。

20.4.4 MIRRN 模型

MIRRN（Multi-granularity Interest Retrieval and Refinement Network）模型是华为提出的长序列用户兴趣建模方法，源自于 KDD 2025 论文《Multi-granularity Interest Retrieval and Refinement Network for Long-Term User Behavior Modeling in CTR Prediction》[27]。前面介绍的 SIM、TWIN 和 TWIN-V2 等方法，主要根据当前候选物品构造检索 Query，从用户长期行为中筛选与目标相关的历史行为，再通过 Attention 提取用户兴趣。这种方式能够有效降低长序列建模的计算开销，但其检索过程主要围绕当前目标物品展开，可能更倾向于保留与目标直接相关的行为，而忽略用户近期兴趣以及相对稳定的全局兴趣。为此，MIRRN 模型提出多粒度兴趣检索机制，分别从目标、近期行为和全局行为三个角度构造检索 Query，获取不同粒度的兴趣表示，并进一步对检索结果进行序列精炼和交互建模。

MIRRN 模型的整体网络结构如图20.26所示，主要由 Embedding Layer、Multi-granularity Interest Retrieval Module、Behavior Sequence Refinement Module、Multi-granularity Interest Activation Module 和 Prediction Layer 组成。模型首先将用户历史行为和目标物品转换为 Embedding 表示，然后通过多粒度检索模块分别获取目标感知、局部感知和全局感知的三个行为子序列。接下来，模型利用位置编码和 Fourier Transformer 对检索后的行为进行进一步建模，得到不同粒度的兴趣表示。最后，模型通过目标感知的多头 Attention 自适应地融合不同粒度的兴趣，形成用户最终表示，并与其他特征一起输入 MLP 完成 CTR 预估。

在 Embedding 层，MIRRN 模型将输入特征分为类别特征和数值特征两类，分别通过 One-hot 或 Multi-hot 编

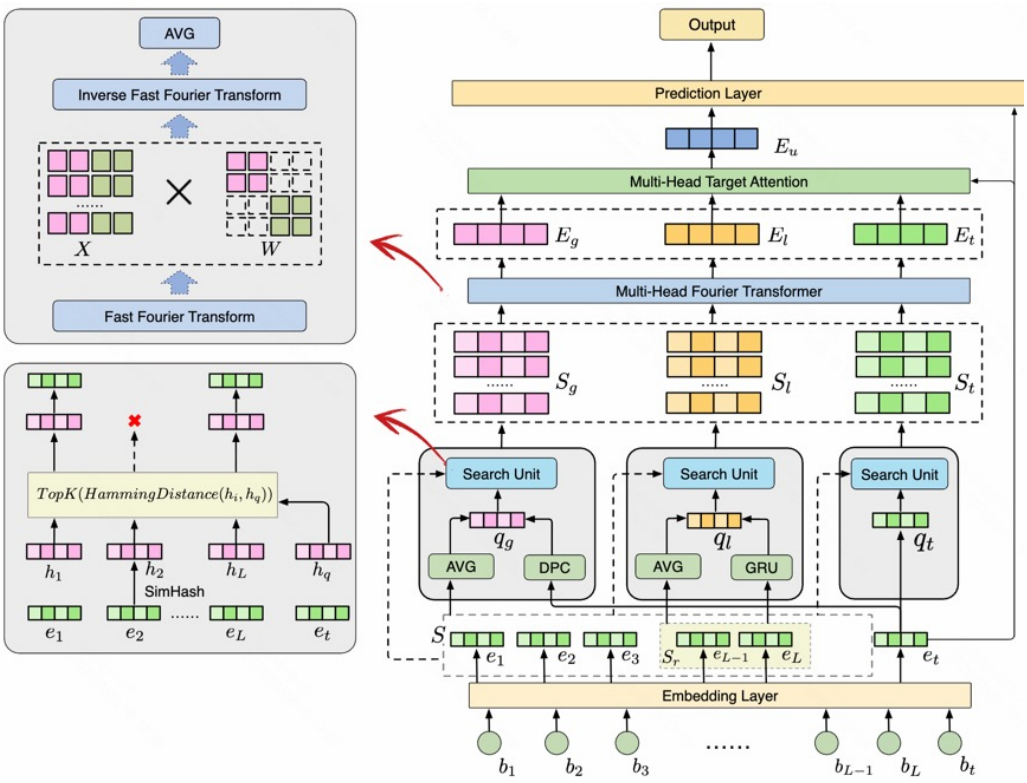


图 20.26: MIRRN 模型的多粒度兴趣检索与融合架构。

码，以及数值分桶等方式转换为稀疏表示，再通过 Embedding 层映射到低维稠密向量空间。假设用户历史行为序列为：

$$\mathbf{S} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_L] \in \mathbb{R}^{L \times d} \tag{20.124}$$

其中，L 表示历史行为序列长度，d 为 Embedding 维度，$\mathbf{e}_i$ 表示第 i 个历史行为的 Embedding。当前目标物品的 Embedding 记为 $\mathbf{e}_t$，其他用户、物品和上下文特征的表示记为 $\mathbf{E}_{\text{other}}$。

MIRRN 模型的第一个核心模块是 Multi-granularity Interest Retrieval Module，其主要目的是从不同时间和兴趣粒度出发构造检索 Query，分别获取与当前目标相关的历史行为、与近期兴趣相关的历史行为，以及能够体现用户整体偏好的历史行为。三个检索单元分别称为 Target-Aware Search Unit（TASU）、Local-Aware Search Unit（LASU）和 Global-Aware Search Unit（GASU），它们使用相同的快速检索机制，但 Query 的构造方式不同。

首先，TASU 直接使用当前目标物品的 Embedding 作为检索 Query：

$$\mathbf{q}_t = \mathbf{e}_t \tag{20.125}$$

根据该 Query，模型从完整历史行为序列中检索出最相关的 K 个行为，得到目标感知的行为子序列：

$$\mathbf{S}_t = \text{Search}(\mathbf{S}, \mathbf{q}_t) \tag{20.126}$$

这种方式与 SIM 和 TWIN 模型中的目标感知检索思路类似，重点保留能够直接反映用户对当前候选物品兴趣的历史行为，因此得到的兴趣粒度相对细致。

在此基础上，MIRRN 模型进一步考虑用户近期行为所反映的局部兴趣，设计了 LASU。与直接使用当前目标物品构造 Query 不同，LASU 使用用户最近一段时间的行为序列构造检索 Query，使其能够体现用户近期的兴趣偏好。假设最近 J 个行为构成的子序列为：

$$\mathbf{S}_r = [\mathbf{e}_{L-J+1}, \ldots, \mathbf{e}_L] \tag{20.127}$$

为了增强近期行为表示的表达能力，论文首先使用 GRU 对该序列进行建模，得到对应的隐藏状态序列 $\mathbf{H}_r$，再

将隐藏状态与原始行为表示分别进行平均池化，得到局部兴趣 Query：

$$\mathbf{H}_r = \mathrm{GRU}(\mathbf{S}_r) \tag{20.128}$$

$$\mathbf{q}_l = \mathrm{Avg}(\mathbf{H}_r) + \mathrm{Avg}(\mathbf{S}_r) \tag{20.129}$$

其中，$\mathrm{Avg}(\cdot)$ 表示平均池化。通过结合 GRU 输出和原始行为 Embedding，Query 既能够保留近期行为的时序信息，也能够利用原始行为的语义信息。随后，LASU 根据 $\mathbf{q}_l$ 从完整历史行为中检索相关行为，得到局部感知的行为子序列：

$$\mathbf{S}_l = \mathrm{Search}(\mathbf{S}, \mathbf{q}_l) \tag{20.130}$$

需要注意的是，LASU 并不是直接将最近 J 个行为作为局部兴趣表示，而是利用近期行为构造 Query，再从整个历史序列中重新检索相关行为。这样既能够利用近期兴趣指导检索，也可以找回较早发生但与当前局部兴趣仍然相关的历史行为，同时避免偶然点击等噪声行为直接主导兴趣表示。

除了当前目标和近期行为之外，MIRRN 模型还希望利用用户整个历史行为序列提取相对稳定的全局兴趣。直接对数千甚至上万条历史行为进行平均池化，容易受到长序列噪声的影响，而使用复杂的序列模型又会增加 Query 构造的计算开销。因此，GASU 引入 Density Peak Clustering（DPC）对预训练物品 Embedding 进行聚类，并利用当前目标物品对应的聚类中心构造全局兴趣 Query。

MIRRN 模型首先对物品集合进行聚类，利用密度和到高密度样本的距离识别聚类中心。对于每个物品，计算其局部密度 ρ_i，以及该物品与密度更高的其他物品之间的最小距离 δ_i。通过 $\rho_i\delta_i$ 得到聚类中心的评价分数，并选取得分最高的若干物品作为聚类中心，其余物品则被分配到距离最近的聚类中心。这样可以得到预先构建的物品聚类索引。

在实际检索时，模型根据当前目标物品的 Embedding $\mathbf{e}_t$，找到与其最近的聚类中心，并将该中心对应的 Embedding 与用户完整历史行为的平均表示相加，得到全局兴趣 Query：

$$\mathbf{q}_g = \mathrm{DPC}(\mathbf{e}_t) + \mathrm{Avg}(\mathbf{S}) \tag{20.131}$$

其中，$\mathrm{DPC}(\mathbf{e}_t)$ 表示当前目标物品对应的最近聚类中心表示。随后，GASU 利用该 Query 从完整历史行为中检索得到全局感知的行为子序列：

$$\mathbf{S}_g = \mathrm{Search}(\mathbf{S}, \mathbf{q}_g) \tag{20.132}$$

由于物品对应的聚类中心可以提前计算并建立索引，在线阶段只需要查询当前目标物品所属的聚类中心即可获得相应表示。通过这种方式，GASU 将目标物品的语义信息与用户整体行为分布结合起来，构造能够体现全局兴趣的检索 Query。对于历史行为较少的新用户，预训练物品 Embedding 和聚类中心也能够提供一定的先验语义信息，从而在一定程度上缓解用户冷启动问题。

上述三个检索单元虽然构造 Query 的方式不同，但均采用 SimHash 实现快速行为搜索。SimHash 是一种局部敏感哈希方法，其基本思想是将高维 Embedding 映射为较短的二进制签名，使得原始向量之间越相似，其哈希签名通常也越相似。关于局部敏感哈希算法，我们在本书第18章中进行过详细介绍，这里不再赘述。

具体来说，MIRRN 模型首先生成固定的随机映射矩阵 $\mathbf{H} \in \mathbb{R}^{d \times m}$，其中 m 为哈希签名的维度。对于任意历史行为 Embedding $\mathbf{e}_i$ 和检索 Query $\mathbf{q}$，分别计算其二进制签名：

$$\mathbf{h}_i = \mathrm{ReLU}\left(\mathrm{Sign}(\mathbf{e}_i\mathbf{H})\right) \tag{20.133}$$

$$\mathbf{h}_q = \mathrm{ReLU}\left(\mathrm{Sign}(\mathbf{q}\mathbf{H})\right) \tag{20.134}$$

其中，$\mathrm{Sign}(\cdot)$ 表示符号函数，$\mathrm{ReLU}(\cdot)$ 用于将结果转换为二进制表示。通过计算历史行为和 Query 对应哈希签名之间的汉明距离（Hamming Distance），即可判断二者的近似相似程度。模型选取汉明距离最小的 K 个历史行为，得到对应的行为索引：

$$\mathbf{I} = \mathrm{TopK}\left(\mathrm{HammingDistance}(\mathbf{h}_i, \mathbf{h}_q)\right) \tag{20.135}$$

最终，通过 Gather 操作从原始行为序列中提取相应的 Embedding，得到检索后的行为子序列：

$$\mathbf{S}^* = \mathrm{Gather}(\mathbf{S}, \mathbf{I}) \tag{20.136}$$

与直接计算高维向量之间的内积相比，二进制签名之间的汉明距离可以利用 XOR 等位运算高效计算。单次签名的距离计算可以近似视为 $O(1)$，显著降低了超长历史行为的检索开销。此外，历史行为对应的哈希签名可以预先计算和存储，在线阶段只需要生成当前 Query 的哈希签名，再完成汉明距离比较和 Top-K 筛选。通过相同的检索机制，三个单元分别得到目标感知、局部感知和全局感知的行为子序列 $\mathbf{S}_t$、$\mathbf{S}_l$ 和 $\mathbf{S}_g$。

在完成多粒度行为检索之后，MIRRN 模型并没有直接对三个行为子序列进行平均池化或简单的 Attention 聚合，而是进一步设计了 Behavior Sequence Refinement Module，对检索结果中的时序信息和行为交互关系进行建模。其主要原因在于，检索过程可能按照相关性对历史行为进行筛选，原始行为的时间顺序容易在这一过程中被弱化，而行为发生的先后关系对于理解用户兴趣的变化仍然具有重要价值。因此，MIRRN 模型首先通过 Target-Aware Position Encoding 恢复行为相对于当前目标的位置信息，再使用 Multi-Head Fourier Transformer 建模不同行为之间的交互关系。

对于三个检索得到的行为子序列，统一记为 $\mathbf{S}^* \in \mathbb{R}^{K\times d}$。MIRRN 模型根据检索得到的原始行为索引，计算每个行为相对于当前目标物品的位置，并通过 Target-Aware Position Encoding（TAPE）查找对应的位置 Embedding，并加入到行为表示中：

$$\mathbf{S}^* = \mathbf{S}^* + \text{Lookup}(\text{TAPE}, \mathbf{I}') \tag{20.137}$$

其中，$\mathbf{I}'$ 表示将原始历史行为索引转换为相对于目标物品的位置索引，Lookup$(\cdot)$ 表示位置 Embedding 的查找操作。与按照历史行为从早到晚的顺序进行位置编码不同，TAPE 以当前目标物品为参考点，使模型能够明确区分不同行为距离当前预测目标的相对位置，从而保留检索后行为序列中的时序信息。

在补充完位置编码之后，MIRRN 模型使用 Multi-Head Fourier Transformer（MHFT）进一步建模检索行为之间的交互。传统 Self-Attention 需要计算历史行为之间的两两相关性，其计算复杂度随序列长度呈平方级增长。虽然经过检索后的行为序列已经明显缩短，但当需要同时建模多个粒度的行为子序列时，仍然有必要进一步降低序列交互的计算开销。因此，MIRRN 模型借鉴 Fourier Transformer 的设计，利用傅里叶变换将行为序列从时域映射到频域，再通过可学习的参数变换频域表示，最后使用逆傅里叶变换恢复行为序列。

MIRRN 模型首先沿着序列长度维度对行为表示执行快速傅里叶变换（FFT），得到频域表示：

$$\mathbf{X} = \mathcal{F}(\mathbf{S}^*) \tag{20.138}$$

其中，$\mathcal{F}(\cdot)$ 表示傅里叶变换。根据卷积定理，频域中的乘法运算对应时域中的循环卷积，因此可以利用较低的计算开销实现序列中不同位置之间的信息交互。基础 Fourier Transformer 通过可学习的复数参数矩阵与频域表示进行逐元素交互，再通过逆傅里叶变换得到新的行为表示。

在此基础上，MIRRN 模型进一步设计 Multi-Head Fourier Transformer，将特征维度拆分为多个子空间，分别学习不同子空间中的行为交互关系。模型首先将频域表示 $\mathbf{X}$ 划分为 n 个子空间：

$$\mathbf{X} = [\mathbf{X}_1; \mathbf{X}_2; \ldots; \mathbf{X}_n] \tag{20.139}$$

其中，$\mathbf{X}_i \in \mathbb{C}^{K\times d/n}$ 表示第 i 个子空间的频域表示。每个子空间通过独立的可学习矩阵 $\mathbf{W}_i$ 进行变换，并结合非线性激活函数得到更新后的表示：

$$\widetilde{\mathbf{X}}_i = \sigma(\mathbf{W}_i \mathbf{X}_i) \tag{20.140}$$

其中，$\mathbf{W}_i \in \mathbb{C}^{d/n\times d/n}$ 为对应子空间的可学习参数，$\sigma(\cdot)$ 表示非线性激活函数。不同子空间之间采用独立的变换参数，以便从不同特征维度提取行为之间的交互信息。

随后，模型将各个子空间的结果重新拼接，并通过逆傅里叶变换恢复时域表示，结合残差连接和 Layer Normalization 得到最终的序列表示：

$$\widetilde{\mathbf{S}}^* = \text{LayerNorm}\left(\mathbf{S}^* + \mathcal{F}^{-1}\left(\text{Concat}(\widetilde{\mathbf{X}}_1, \ldots, \widetilde{\mathbf{X}}_n)\right)\right) \tag{20.141}$$

与传统 Multi-Head Self-Attention 相比，MHFT 不需要计算序列中所有行为之间的两两注意力权重，而是通过频域变换和多个特征子空间的交互来提取行为关联信息。其计算复杂度为 $O\left(Kd\log K + \frac{Kd^2}{n}\right)$，其中第一项来自 FFT 和 IFFT 操作，第二项来自多个子空间中的特征变换。相比起传统 Multi-Head Self-Attention 的 $O(K^2d + Kd^2)$ 时间复杂度，MHFT 能够以较低的计算开销建模检索后行为之间的交互关系。

在经过 TAPE 和 MHFT 的处理后，MIRRN 模型得到精炼后的行为序列表示，再通过平均池化得到对应粒度的用户兴趣向量：

$$\mathbf{E}^* = \mathrm{Avg}(\widetilde{\mathbf{S}}^*) \tag{20.142}$$

分别对三个粒度的行为子序列进行上述处理，可以得到目标感知、局部感知和全局感知的三个兴趣向量，分别记为 $\mathbf{E}_t$、$\mathbf{E}_l$ 和 $\mathbf{E}_g$。这样，模型不仅保留了不同粒度的兴趣信息，还进一步融合了检索行为之间的时序关系和交互信息。

在获得三个粒度的兴趣向量之后，MIRRN 模型进一步设计 Multi-granularity Interest Activation Module，根据当前目标物品动态调整不同粒度兴趣的贡献。由于用户对不同候选物品的兴趣可能存在明显差异，目标感知、局部感知和全局感知兴趣在不同预测任务中的重要性也不相同，因此需要通过目标感知的 Attention 进行自适应融合。

模型首先将三个兴趣向量拼接为矩阵：

$$\mathbf{E}_m = [\mathbf{E}_t; \mathbf{E}_l; \mathbf{E}_g] \in \mathbb{R}^{3\times d} \tag{20.143}$$

随后，模型以当前目标物品 Embedding $\mathbf{e}_t$ 作为 Query，将三个粒度的兴趣向量分别映射为 Key 和 Value。对于第 i 个 Attention Head，其计算过程为：

$$\mathbf{Q}_i = \mathbf{W}_i^Q \mathbf{e}_t, \qquad \mathbf{K}_i = \mathbf{W}_i^K \mathbf{E}_m, \qquad \mathbf{V}_i = \mathbf{W}_i^V \mathbf{E}_m \tag{20.144}$$

$$\mathrm{head}_i = \mathrm{softmax}\left(\frac{\mathbf{Q}_i \mathbf{K}_i^\top}{\sqrt{d}}\right)\mathbf{V}_i \tag{20.145}$$

将多个 Attention Head 的输出进行拼接和线性映射，就可以得到融合后的用户兴趣表示：

$$\mathbf{E}_u = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h)\mathbf{W}^O \tag{20.146}$$

其中，h 表示 Attention Head 的数量，$\mathbf{W}_i^Q$、$\mathbf{W}_i^K$、$\mathbf{W}_i^V$ 和 $\mathbf{W}^O$ 均为可学习参数。通过这一机制，模型能够根据当前目标物品，自适应地分配三种不同粒度兴趣的权重，使最终用户表示同时包含目标相关的细粒度兴趣、近期兴趣以及更加稳定的全局兴趣。

最后，MIRRN 模型将目标物品 Embedding $\mathbf{e}_t$、融合后的用户兴趣表示 $\mathbf{E}_u$ 以及其他用户、物品和上下文特征 $\mathbf{E}_{\mathrm{other}}$ 拼接，输入 MLP 完成 CTR 预估：

$$\hat{y} = \mathrm{Sigmoid}\left(\mathrm{MLP}\left([\mathbf{e}_t; \mathbf{E}_u; \mathbf{E}_{\mathrm{other}}]\right)\right) \tag{20.147}$$

MIRRN 模型采用交叉熵损失对上述的 CTR 预估结果 $\hat{y}$ 进行端到端训练。

总的来说，MIRRN 模型在传统目标感知检索的基础上，将用户兴趣建模扩展到目标、局部和全局三个不同粒度。多粒度检索模块通过构造不同的 Query，从完整历史行为中获取具有不同兴趣侧重点的行为子序列。行为序列精炼模块利用目标感知位置编码和 Fourier Transformer 恢复时序信息并建模行为交互。最后，兴趣激活模块根据当前目标物品自适应融合不同粒度的兴趣表示。相比起 SIM 和 TWIN 模型主要围绕当前候选物品进行检索的方式，MIRRN 模型进一步利用近期行为和全局行为信息补充目标感知兴趣，使模型能够从不同时间范围和兴趣粒度理解用户偏好，同时通过 SimHash 和 MHFT 控制长序列检索及建模的计算开销。

20.4.5 FIM 模型

FIM（Frequency-Aware Multi-View Interest Modeling）模型是快手提出的面向本地生活服务推荐的用户兴趣建模方法，源自 SIGIR 2025 论文《FIM: Frequency-Aware Multi-View Interest Modeling for Local-Life Service Recommendation》[22]。与前面介绍的 SIM、TWIN 等主要围绕目标物品进行相关行为检索的方法不同，FIM 模型进一步考虑了本地生活服务场景中用户兴趣的多维属性以及周期性变化特点。例如，用户对本地生活服务的消费偏好不仅与商品类别有关，还可能受到品牌、价格和作者等因素的影响。同时，不同用户的消费行为也可能呈现重复性、季节性或者探索性的周期模式。只依靠目标物品与历史行为之间的语义相关性进行检索，或者直接在时域对行为序列进行建模，可能难以充分捕获这些复杂的兴趣特征。因此，FIM 模型将多视角行为检索与频域兴趣建模结合起来，从不同物品属性和不同频率成分中提取用户兴趣。

FIM 模型的整体网络结构如图20.27所示，主要包含多视角检索、多视角兴趣建模、频域感知与演化以及预测四个部分。首先，模型根据目标物品的作者、品牌、品类和价格等属性，从用户完整历史行为中检索多个相关行为子序列。随后，模型利用 Target Attention 分别提取不同视角下的用户兴趣。同时，模型将历史行为 Embedding 转换到频域，通过不同频率范围的过滤提取用户行为中的长期变化趋势、周期性变化和快速波动信息，并利用用户及目标物品特征自适应地调整不同频率成分的贡献。最后，模型将多视角兴趣表示与频域兴趣表示进行融合，通过 MMoE 完成点击率、观看时长和购买率等多任务预估。

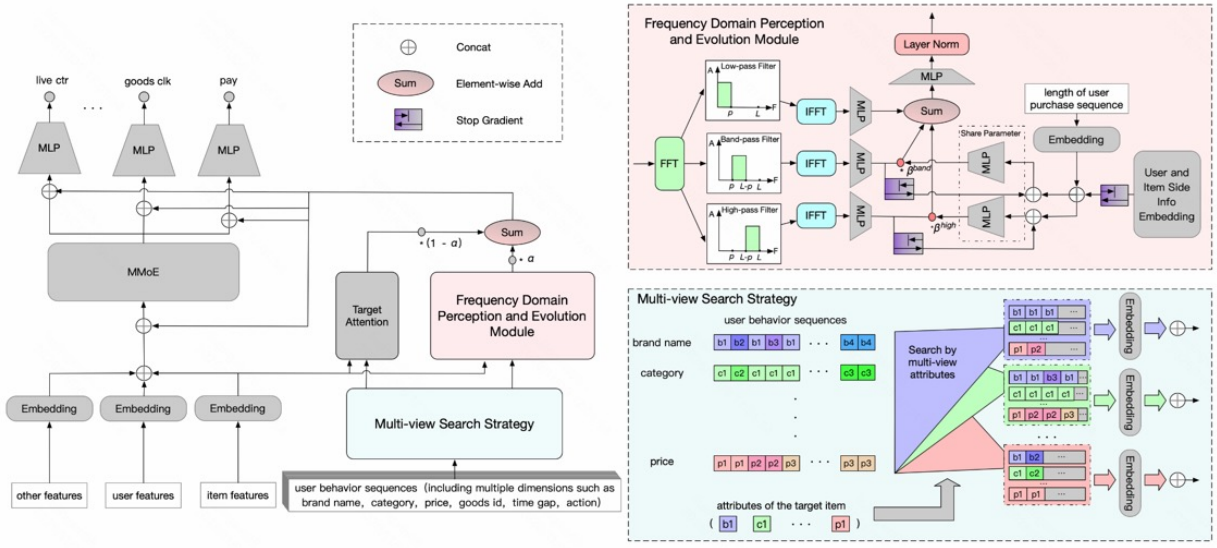


图 20.27: FIM 模型的多视角兴趣检索与频域建模架构。

首先，在多视角兴趣建模阶段，FIM 模型针对本地生活服务场景中用户行为的多维属性，设计了 Multi-view Search Strategy（MSS），从不同属性视角检索与当前目标物品相关的历史行为。快手本地生活场景的用户行为包含商品 ID、作者 ID、来源域、行为类型、品牌、商品类别、时间跨度、商品价格和支付金额等九类属性。这些属性从不同角度反映了用户的消费偏好，而不同属性对于用户当前购买决策的重要性也可能存在差异。因此，FIM 模型选择目标物品的作者 ID、品牌、商品类别和价格作为四个检索视角，分别构建独立的行为检索过程。

对于第 v 个检索视角，假设历史行为物品和当前目标物品在该视角下对应的属性分别为 m_i^v 和 m_t^v，其相关性得分可以通过 Hard Search 或 Soft Search 两种方式计算：

$$r_i^v = \begin{cases} \mathbb{I}(m_i^v = m_t^v), & \text{Hard Search} \\ \langle W_b \mathbf{e}_i^v, W_t \mathbf{e}_t^v \rangle, & \text{Soft Search} \end{cases} \tag{20.148}$$

其中，$\mathbf{e}_i^v$ 和 $\mathbf{e}_t^v$ 分别表示历史行为物品和目标物品在对应视角下的 Embedding，W_b 和 W_t 为可学习的映射矩阵，$\langle \cdot, \cdot \rangle$ 表示向量内积。对于 Hard Search，模型直接根据两个物品在相应属性上的匹配关系进行筛选；对于 Soft Search，则通过可学习的 Embedding 交互计算二者之间的语义相关性。随后，模型根据相关性得分选取 Top-K 个历史行为，形成对应视角下的行为子序列：

$$\mathcal{S}_v = \text{TopK}\left(\mathcal{S}, \{r_i^v\}_{i=1}^N\right) \tag{20.149}$$

通过上述方式，FIM 模型可以分别获得作者、品牌、品类和价格四个视角下的历史行为子序列。与 SIM 模型主要根据目标物品的类别或者 Embedding 相关性检索历史行为不同，FIM 模型将物品的多个属性作为独立检索视角，使模型能够从不同角度挖掘用户的消费偏好。例如，用户可能对某个品牌具有稳定偏好，也可能对某个价格区间的商品表现出更高的购买意愿，多视角检索能够将这些不同维度的兴趣信息分别保留下来。

在获得不同视角的行为子序列之后，FIM 模型使用 Target Attention 进一步提取与当前目标物品相关的兴趣表示。模型将当前目标物品的特征作为 Query，将各个视角检索得到的多维行为特征作为 Key 和 Value，分别计算目标物品与历史行为之间的注意力权重，并对行为表示进行加权聚合。最终，模型将四个视角的 Attention 输

出进行拼接，得到多视角兴趣表示：

$$\mathbf{H}_u = \text{Concat}\left(\mathbf{H}_u^1, \mathbf{H}_u^2, \mathbf{H}_u^3, \mathbf{H}_u^4\right) \tag{20.150}$$

其中，$\mathbf{H}_u^v$ 表示第 v 个属性视角下通过 Target Attention 得到的用户兴趣表示。通过这种方式，模型能够分别学习不同物品属性所对应的用户兴趣，并保留不同视角之间的互补信息。

除了从不同物品属性中提取用户兴趣，FIM 模型还进一步考虑用户行为在时间维度上的周期性变化。尤其是在本地生活服务场景中，用户的消费行为可能具有明显的周期性，例如重复购买日常用品、特定季节的消费需求，或者受到短期促销活动影响而产生的集中购买行为。这些模式在原始时域行为序列中可能与大量其他行为混合在一起，难以直接识别。因此，FIM 模型设计了 Frequency domain Perception and Evolution Module（FPEM），将用户历史行为序列从时域转换到频域，并通过不同频率范围的过滤操作提取多种时间尺度上的兴趣信息。

假设用户行为 Embedding 序列为 $\mathbf{E}_u \in \mathbb{R}^{N\times D}$，其中 N 为历史行为长度，D 为 Embedding 维度。FIM 模型沿序列维度执行离散傅里叶变换（Discrete Fourier Transform，DFT），将原始行为表示转换为频域表示：

$$\mathbf{F}_u = \mathcal{F}(\mathbf{E}_u) \tag{20.151}$$

其中，$\mathcal{F}(\cdot)$ 表示傅里叶变换。由于实数输入经过傅里叶变换后具有共轭对称性，因此只需要保留一半频谱即可完成后续建模。在实际计算中，FIM 模型使用 Fast Fourier Transform（FFT）代替直接计算 DFT，将计算复杂度由 $O(N^2)$ 降低到 $O(N\log N)$。通过频域表示，用户行为序列中的不同变化速度可以被分离到不同的频率范围，为识别用户兴趣的周期性变化提供了基础。

在获得频谱之后，FIM 模型进一步设计了低通滤波器、带通滤波器和高通滤波器，分别提取低频、中频和高频成分。假设频谱长度为 $L = \lceil N/2\rceil + 1$，p 为控制频率范围的超参数，则三种滤波器可以分别表示为：

$$\text{filter}_{\text{low}}(c) = \begin{cases} 1, & c \in [0, p) \\ 0, & \text{otherwise} \end{cases} \tag{20.152}$$

$$\text{filter}_{\text{band}}(c) = \begin{cases} 1, & c \in [p, L-p) \\ 0, & \text{otherwise}, \end{cases} \tag{20.153}$$

$$\text{filter}_{\text{high}}(c) = \begin{cases} 1, & c \in [L-p, L) \\ 0, & \text{otherwise} \end{cases} \tag{20.154}$$

其中，c 表示频率分量的索引。低通滤波器保留变化较为缓慢的低频成分，主要用于提取用户长期且相对稳定的消费偏好；带通滤波器保留中间频率范围的成分，用于捕获介于长期趋势和快速波动之间的行为模式；高通滤波器则保留变化较快的高频成分，能够反映用户短期兴趣波动以及由促销等因素引发的阶段性消费行为。

通过将频谱分别与三种滤波器相乘，再执行逆快速傅里叶变换（IFFT），模型可以重新得到时域下的低频、中频和高频行为表示：

$$\mathbf{f}_u^{\text{low}}, \mathbf{f}_u^{\text{band}}, \mathbf{f}_u^{\text{high}} = \mathcal{F}^{-1}\left(\text{filter}_{\text{low}}(\mathbf{F}_u), \text{filter}_{\text{band}}(\mathbf{F}_u), \text{filter}_{\text{high}}(\mathbf{F}_u)\right) \tag{20.155}$$

由于高频和中频成分中可能包含较多短期噪声和偶然行为，直接将三种频率成分等权融合可能引入不必要的干扰。因此，FIM 模型进一步设计了频率感知机制，根据用户和目标物品的侧信息，自适应地调整中频和高频成分的贡献。模型首先将用户及目标物品的侧信息表示记为 $\mathbf{I}_u$，其中包含用户画像、目标物品的品牌和价格等特征，以及用户历史购买行为序列长度等信息。随后，模型将 $\mathbf{I}_u$ 与对应频率成分进行拼接，通过两层感知机计算自适应门控权重：

$$\boldsymbol{\beta}_{\text{band}} = \sigma\left(W_2\phi\left(W_1[\mathbf{I}_u; \mathbf{f}_u^{\text{band}}] + B_1\right) + B_2\right) \tag{20.156}$$

$$\boldsymbol{\beta}_{\text{high}} = \sigma\left(W_2\phi\left(W_1[\mathbf{I}_u; \mathbf{f}_u^{\text{high}}] + B_1\right) + B_2\right) \tag{20.157}$$

其中，$\phi(\cdot)$ 和 $\sigma(\cdot)$ 分别表示 ReLU 和 Sigmoid 激活函数，W_1、W_2、B_1 和 B_2 为可学习参数。通过这一机制，模型可以结合不同用户及目标物品的特征，动态判断中频和高频信息对当前预测任务的重要性。

最终，FIM 模型将低频成分作为基础表示，并利用上述门控权重对中频和高频成分进行加权融合：

$$\mathbf{f}_u^{\text{fuse}} = \mathbf{f}_u^{\text{low}} + \boldsymbol{\beta}_{\text{band}} \odot \mathbf{f}_u^{\text{band}} + \boldsymbol{\beta}_{\text{high}} \odot \mathbf{f}_u^{\text{high}} \tag{20.158}$$

其中，$\odot$ 表示逐元素相乘。随后，通过残差连接将融合后的频域表示与原始行为 Embedding 相加，并利用 Layer Normalization 得到最终的频域兴趣表示：

$$\mathbf{f}_u = \text{LayerNorm}\left(\mathbf{f}_u^{\text{fuse}} + \mathbf{E}_u\right) \tag{20.159}$$

通过上述设计，FIM 模型不仅能够区分用户行为中的不同频率成分，还能够根据用户和目标物品的特征自适应地控制不同频率信息的贡献，从而同时保留长期稳定的行为趋势以及对近期兴趣变化的感知能力。与直接在时域使用 Self-Attention 建模行为之间的两两关系相比，FPEM 通过 FFT、频率过滤和 IFFT 提取序列中的周期性信息，其主要计算复杂度为 $O(N \log N + ND^2)$，其中 $O(N \log N)$ 来自傅里叶变换，$O(ND^2)$ 主要来自频率感知模块中的 MLP 运算。相比传统 Self-Attention 在序列维度上的 $O(N^2 D)$ 计算开销，FPEM 在长序列场景下具有更好的计算效率。

在完成多视角兴趣建模和频域兴趣建模之后，FIM 模型需要将两部分结果融合，形成最终的用户兴趣表示。由于不同用户的历史行为长度不同，模型首先分别沿行为序列维度进行平均池化，得到多视角兴趣表示 $\overline{\mathbf{H}}_u$ 和频域兴趣表示 $\bar{\mathbf{f}}_u$，然后通过加权求和实现两路兴趣信息的融合：

$$\mathbf{Z}_u = (1-\alpha)\overline{\mathbf{H}}_u + \alpha\bar{\mathbf{f}}_u \tag{20.160}$$

其中，α 为控制两种兴趣表示相对重要性的超参数。最终，FIM 模型将融合后的用户兴趣表示与其他特征输入 MMoE（Multi-gate Mixture-of-Experts）进行多任务学习，通过不同任务对应的门控网络，自适应地组合共享专家网络的输出，并利用任务专属的预测头分别完成点击率、观看时长和购买率等指标的预估。此外，模型还通过残差连接增强特征表示能力，并采用相应的交叉熵损失对预测任务进行优化。

总体而言，FIM 模型在传统基于检索的用户行为序列建模方法上进一步扩展了兴趣提取的维度。一方面，通过作者、品牌、品类和价格四个视角进行行为检索，模型能够从不同物品属性中提取更加全面的用户消费偏好。另一方面，通过频域变换和多频率过滤，模型能够识别用户历史行为中不同时间尺度的变化模式，并根据用户及目标物品特征自适应融合不同频率成分。与 SIM、TWIN 等主要关注目标相关行为检索的方法相比，FIM 模型将多视角属性检索与频域兴趣演化结合起来，尤其适用于用户消费行为具有多维属性和周期性变化特征的本地生活服务推荐场景。

20.4.6 LCN 模型

LCN（Lifelong Cross Network）是微信团队提出的跨域终身用户行为建模方法，源自 KDD 2024 论文《Cross-Domain LifeLong Sequential Modeling for Online Click-Through Rate Prediction》[14]。该模型针对传统终身行为建模主要利用单一领域历史行为的问题，进一步考虑了用户在不同业务领域之间的兴趣关联，通过源域（Source Domain）的长期行为序列增强目标域（Target Domain）的 CTR 预估。与 SIM、TWIN 等主要针对同一领域的长期历史行为进行目标感知检索的方法不同，LCN 模型面临的是更加复杂的**跨域建模问题**：

笔记 源域和目标域具有相同的用户群体，但两个领域中的物品不存在重叠，用户在不同领域中的行为分布和交互模式也可能存在较大差异。因此，如何建立不同领域物品之间的语义联系，并从源域超长行为序列中准确提取与目标物品相关的用户兴趣，是跨域终身行为建模需要解决的核心问题。

LCN 模型主要由 Cross Representation Production（CRP）和 Lifelong Attention Pyramid（LAP）两个模块组成。CRP 通过对比学习增强不同领域物品 Embedding 之间的关联，使源域行为能够更好地匹配目标域候选物品；LAP 则通过三级级联注意力，逐步筛选与目标物品相关的历史行为，并在逐渐缩小的候选集合上进行更加精细的兴趣建模。LCN 模型的整体网络结构如图20.28所示。模型以用户画像、源域和目标域的短期行为序列以及源域的终身行为序列作为输入，其中短期行为序列主要用于 CRP 的跨域表示学习，源域终身行为序列则通过 LAP 提取长期兴趣。最终，模型将得到的用户兴趣表示与其他特征结合，完成目标域 CTR 预估。

首先，CRP 模块主要解决跨域物品表示不一致的问题。在跨域终身行为建模中，用户的源域历史行为和当前目标物品来自不同业务领域，由于两个领域中的物品不存在重叠，仅依靠目标域 CTR 任务进行训练，很难保

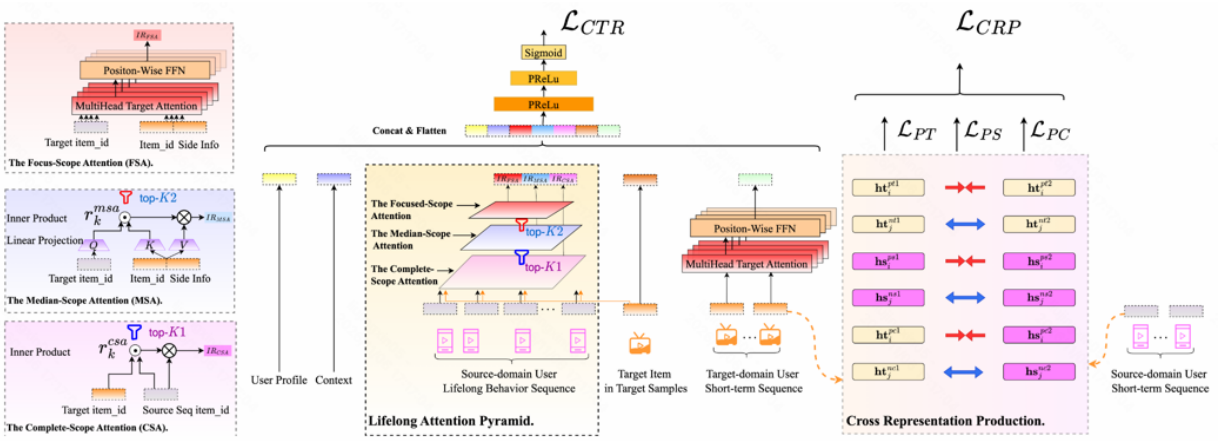


图 20.28: LCN 模型的跨域表示学习与三级注意力建模架构。

证源域和目标域的物品 Embedding 具有一致的语义空间。如果直接利用这些 Embedding 计算物品之间的相关性，可能导致模型无法准确识别跨域行为中的潜在兴趣关联，影响后续长期行为检索的效果。

因此，LCN 模型借鉴对比学习的思想，利用用户在两个领域中的短期行为构造正负样本对，额外监督不同领域物品 Embedding 的学习。其基本假设如下：

笔记 同一个用户在短时间内的兴趣具有一定的跨域稳定性，因此其在不同领域发生的交互行为可能反映相似的潜在兴趣。

根据这一观察，CRP 设计了三种正样本对。第一种是**目标域内部的正样本对（Positive Target Pair）**，从同一个用户的目标域短期行为序列中随机选取两个物品，使其 Embedding 在表示空间中更加接近。第二种是**源域内部的正样本对（Positive Source Pair）**，从同一个用户的源域短期行为序列中选取两个物品，以增强源域内部物品表示的一致性。第三种是**跨域正样本对（Positive Cross-domain Pair）**，分别从同一个用户的目标域和源域短期行为序列中选取物品，约束两个领域中反映相似用户兴趣的物品 Embedding 相互接近。

与正样本对相对应，CRP 还构造了三类负样本对。其构造方式与正样本类似，但从当前 Mini-batch 中不同用户的行为序列中选取物品，使不同用户的物品表示在相应的域内和跨域关系上保持区分。这样，模型能够同时学习目标域内部、源域内部以及跨域物品之间的表示关系，而不是仅依赖某一个领域的物品语义。图20.29展示了 CRP 的样本构造方式。通过三类正负样本对，模型分别约束同一领域内的兴趣表示以及不同领域之间的语义关联，从而为后续源域历史行为与目标物品之间的相关性计算提供更加一致的 Embedding 表示。

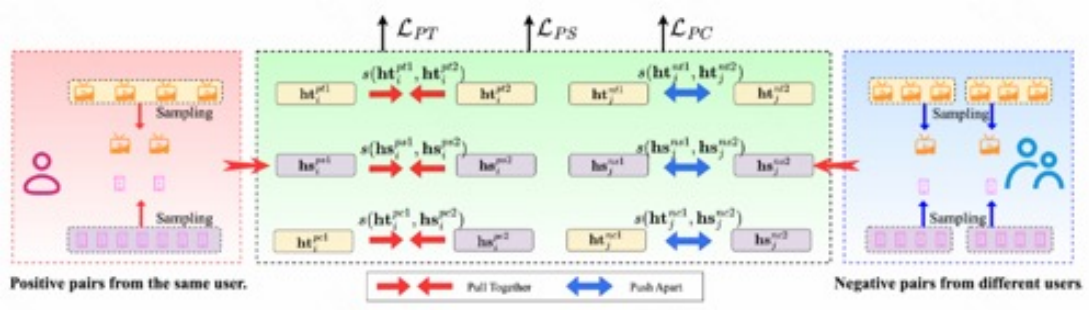


图 20.29: CRP 模块的跨域正负样本构造与对比学习机制。

假设 $s(\cdot,\cdot)$ 表示两个物品 Embedding 之间的余弦相似度，τ 为控制相似度分布的温度系数。对于三种正样本对及其对应的负样本对，分别构造对比学习损失。以目标域内部的正样本对为例，其损失可以表示为：

$$\mathcal{L}_{PT} = -\frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\log\frac{\exp(s(\mathbf{h}^t_{pt1,i},\mathbf{h}^t_{pt2,i})/\tau)}{\sum_{j=1}^{M}\exp(s(\mathbf{h}^t_{nt1,j},\mathbf{h}^t_{nt2,j})/\tau)} \tag{20.161}$$

其中，$\mathbf{h}^t_{pt1,i}$ 和 $\mathbf{h}^t_{pt2,i}$ 表示第 i 个目标域正样本对的物品 Embedding，$\mathbf{h}^t_{nt1,j}$ 和 $\mathbf{h}^t_{nt2,j}$ 表示第 j 个目标域负样本对的物品 Embedding，M 为当前 Batch 中采样的负样本对数量。类似地，可以构造源域内部的对比学习损失和跨

域对比学习损失，分别记为 $\mathcal{L}_{PS}$ 和 $\mathcal{L}_{PC}$。最终，CRP 的整体损失为：

$$\mathcal{L}_{CRP} = \lambda_{PT}\mathcal{L}_{PT} + \lambda_{PS}\mathcal{L}_{PS} + \lambda_{PC}\mathcal{L}_{PC} \tag{20.162}$$

其中，λ_{PT}、λ_{PS} 和 λ_{PC} 分别用于控制三类对比学习目标的权重。通过上述方式，CRP 能够在保持各领域内部物品表示能力的同时，进一步增强跨域物品之间的语义联系。由于 CRP 与主 CTR 预估网络联合训练，其学习得到的跨域物品表示也能够持续受到目标域预测任务的反馈。

在解决跨域物品表示问题之后，LCN 模型进一步通过 LAP 模块从源域终身行为序列中提取与当前目标物品相关的用户兴趣。传统的终身序列建模通常采用两阶段结构，包括 General Search Unit（GSU）和 Exact Search Unit（ESU）：第一阶段利用相对简单的相关性计算筛选历史行为，第二阶段再通过更复杂的 Attention 提取精细兴趣。这种方式能够有效降低计算开销，但由于第一阶段与第二阶段使用的注意力结构和相关性度量存在差异，初步筛选结果可能与后续精细建模真正关注的行为不完全一致。这一问题在跨域场景下更加明显。源域和目标域的物品及其交互模式存在较大差异，源域历史行为中可能包含更多与当前目标无关的信息。如果第一阶段仅使用简单的相关性度量，部分具有潜在跨域兴趣价值的行为可能在早期就被过滤掉。

因此，LCN 模型将传统两阶段结构进一步扩展为三级级联注意力，构建 Lifelong Attention Pyramid。该模块由 Complete-Scope Attention（CSA）、Median-Scope Attention（MSA）和 Focused-Scope Attention（FSA）组成，三个层级依次处理完整历史行为、初步筛选后的行为以及进一步精简后的行为集合。随着候选行为数量逐渐减少，模型可以逐步引入更加丰富的上下文特征和复杂的注意力结构，在保证计算效率的同时提高兴趣提取的准确性。

首先，CSA 作为第一级注意力，负责对完整的源域终身行为序列进行初步筛选。由于此时需要处理的历史行为数量较大，直接采用复杂的 Multi-head Attention 会带来较大的计算开销，因此 CSA 使用简单的内积相关性计算目标物品与历史行为之间的匹配程度。假设当前目标物品 Embedding 为 $\mathbf{e}_i^v$，源域历史行为中第 k 个物品的 Embedding 为 $\mathbf{e}_k^h$，则对应的相关性得分为：

$$r_k^{CSA} = \langle \mathbf{e}_k^h, \mathbf{e}_i^v \rangle \tag{20.163}$$

其中，$\langle \cdot, \cdot \rangle$ 表示向量内积。随后，根据相关性得分对所有历史行为进行排序，选取 Top-K_1 个相关行为构成下一层的候选子序列。同时，CSA 根据相关性得分对原始行为表示进行加权聚合，得到初步的兴趣表示：

$$\mathbf{IR}_{CSA} = \sum_{k=1}^{N} r_k^{CSA}\mathbf{e}_k^h \tag{20.164}$$

其中，N 为完整终身行为序列的长度，$\mathbf{IR}_{CSA}$ 表示第一级注意力得到的兴趣表示。从上面的推导可以看出，CSA 的主要作用是从完整历史行为中去除明显不相关的物品，并保留一个相对宽泛的候选行为集合，供后续注意力层进一步处理。

在 CSA 的基础上，MSA 作为第二级注意力，进一步利用历史行为的上下文信息对候选集合进行筛选。作者观察到，在终身行为序列中，大量注意力得分往往集中在少数历史行为上，初始阶段的简单相关性筛选难以充分利用行为的交互信息。因此，MSA 在 CSA 输出的 Top-K_1 个行为基础上，引入行为上下文特征，例如观看时长、交互详情等，使模型能够从更丰富的信息中判断历史行为与目标物品之间的相关性。

假设第 k 个历史行为的 Embedding 为 $\mathbf{e}_k^h$，对应的上下文 Embedding 为 $\mathbf{e}_k^c$，首先将二者拼接，再通过线性映射得到 Key 和 Value；当前目标物品则作为 Query。其注意力权重定义为：

$$\alpha_k^{MSA} = \mathrm{softmax}_k\left(\frac{(W^Q\mathbf{e}_i^v)^\top W^K(\mathbf{e}_k^h \| \mathbf{e}_k^c)}{\sqrt{d}}\right) \tag{20.165}$$

其中，W^Q、W^K 和 W^V 分别表示 Query、Key 和 Value 对应的投影矩阵，$\|$ 表示向量拼接，d 为注意力计算的特征维度。MSA 根据注意力权重对候选行为及其上下文表示进行加权聚合，得到第二级兴趣表示：

$$\mathbf{IR}_{MSA} = \sum_{k=1}^{K_1} \alpha_k^{MSA} W^V(\mathbf{e}_k^h \| \mathbf{e}_k^c) \tag{20.166}$$

随后，MSA 按照注意力得分选取 Top-K_2 个行为，作为最后一级 FSA 的输入。通过增加中间层，模型能够在完整序列的初步筛选和高精度兴趣建模之间建立过渡，使更多上下文信息能够参与历史行为的筛选过程。

最后，FSA 作为第三级注意力，负责从经过前两级筛选的 Top-K_2 个行为中提取更加精细的目标感知兴趣。

由于此时历史行为数量已经明显减少，模型可以使用更复杂的 Multi-head Attention 结构，对不同表示子空间中的用户兴趣进行建模。其基本结构类似于 Transformer Decoder 中的交叉注意力机制，以当前目标物品作为 Query，以历史行为及其上下文特征作为 Key 和 Value。对于第 h 个 Attention Head，计算过程为：

$$\mathbf{H}_h = \text{softmax}\left(\frac{\mathbf{Q}_h\mathbf{K}_h^\top}{\sqrt{d}}\right)\mathbf{V}_h \tag{20.167}$$

其中，$\mathbf{Q}_h = W_h^Q\mathbf{e}_i^v$，$\mathbf{K}_h = W_h^K(\mathbf{e}_k^h \| \mathbf{e}_k^c)$，$\mathbf{V}_h = W_h^V(\mathbf{e}_k^h \| \mathbf{e}_k^c)$ 分别表示第 h 个 Attention Head 的 Query、Key 和 Value。随后，FSA 将多个 Attention Head 的输出进行拼接，并通过前馈网络进一步提取非线性特征：

$$\mathbf{H} = \text{ReLU}\left(\text{Concat}(\mathbf{H}_1, \ldots, \mathbf{H}_l)W_1 + b_1\right)W_2 + b_2 \tag{20.168}$$

其中，l 为 Attention Head 的数量，W_1、W_2、b_1 和 b_2 为前馈网络的可学习参数。最后，对 Top-K_2 个行为对应的输出进行平均池化，可以得到 FSA 的兴趣表示：

$$\mathbf{IR}_{FSA} = \frac{1}{K_2}\sum_{k=1}^{K_2}\mathbf{H}_k \tag{20.169}$$

与前两级相比，FSA 使用了更加复杂的多头注意力和前馈网络，能够对经过逐步筛选的相关历史行为进行更充分的非线性建模。LAP 模块最终将 CSA、MSA 和 FSA 三个层级得到的兴趣表示进行融合，形成综合利用完整行为范围、上下文信息和精细目标相关性的用户兴趣表示。由于三个层级均保留了相应的兴趣输出，模型的训练梯度可以同时通过不同层级传递，避免仅依赖最终筛选结果进行兴趣学习。

在模型训练阶段，LCN 模型将主 CTR 预估损失与 CRP 的跨域对比学习损失联合优化，最终目标函数为：

$$\mathcal{L} = \mathcal{L}_{CTR} + \lambda_{CRP}\mathcal{L}_{CRP} \tag{20.170}$$

其中，$\mathcal{L}_{CTR}$ 为目标域的 CTR 预估损失，$\mathcal{L}_{CRP}$ 为跨域表示学习损失，λ_{CRP} 用于控制辅助对比学习任务的影响。通过端到端联合训练，CRP 学习得到的跨域物品表示能够为 LAP 提供更加一致的行为与目标表示，而 LAP 从源域终身行为中提取的兴趣信息又能够通过主 CTR 任务进一步优化跨域表示学习。

总的来说，LCN 模型从跨域表示对齐和多级兴趣筛选两个方面扩展了传统终身行为建模方法。一方面，CRP 利用目标域、源域以及跨域三类正负样本对进行对比学习，改善不同领域物品 Embedding 之间的语义关联；另一方面，LAP 将传统两阶段检索扩展为三级级联注意力，通过 CSA、MSA 和 FSA 逐步缩小历史行为范围，并在更小的候选集合上使用更加复杂的 Attention 结构提取兴趣。与 SIM 和 TWIN 等显式检索式方法相比，LCN 模型更强调跨域表示学习与级联注意力筛选的结合，尤其适用于用户群体重叠、物品集合不重叠的跨域推荐场景。

20.4.7 LCN 模型的后续应用

在 LCN 模型之后，微信团队进一步研究了如何在长序列建模过程中利用历史行为之间的上下文关系，并提出了 Context-Aware Interest Network（CAIN）模型 [12]。前面介绍的 LCN 模型通过 Lifelong Attention Pyramid（LAP）逐层筛选和聚合长序列中的目标相关行为，解决了从大规模历史行为中高效提取用户兴趣的问题。不过，这类方法在计算注意力时，主要依据历史行为与目标物品之间的相关性来衡量单个行为的重要程度，并没有显式建模该行为在原始序列中的局部上下文。例如，用户连续观看一组短视频时，某个视频虽然获得了较长的观看时长，但它之后的多个视频可能都只获得了很短的观看时长。这一变化可能意味着用户在观看该视频前后发生了兴趣转移，或者该视频本身包含导致用户兴趣下降的内容特征。如果模型只根据单个视频与当前目标物品的相关性评估其价值，就可能忽略相邻行为所反映出的兴趣变化。

因此，CAIN 模型的核心思路如下：

定义 20.5

CAIN 模型在长序列注意力计算之前引入时序卷积网络（Temporal Convolutional Network，TCN），将每个历史行为与其前后相邻行为结合起来，生成包含上下文信息的行为表示，再利用这些表示进行目标感知的兴趣提取。



与 RNN 依赖逐步状态更新、Self-Attention 需要计算序列内部多个位置之间的交互不同，TCN 通过卷积窗口

在序列上进行局部计算，能够并行处理不同位置的行为表示，并且可以直接通过卷积核尺寸控制每个行为所包含的上下文范围。

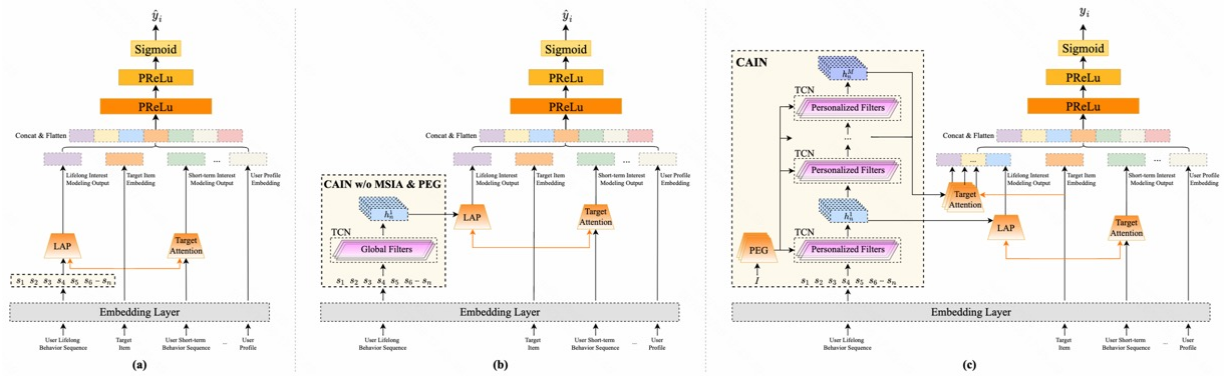


图 20.30: CAIN 模型的网络结构演进。

从整体结构看，CAIN 模型以基于 LAP 的长序列建模网络作为基础，在此基础上逐步引入 TCN、Multi-Scope Interest Aggregator（MSIA）和 Personalized Extractor Generation（PEG）两个模块。图20.30展示了这一结构演进过程：图（a）为采用 LAP 的基线模型；图（b）在原有长序列建模分支中加入 TCN，使进入 LAP 的行为表示具备上下文信息；图（c）则进一步引入 MSIA 和 PEG，分别实现多种上下文范围下的兴趣提取，以及面向不同用户的个性化卷积。

在 CAIN 模型中，用户的输入特征主要包括基础画像特征、短期行为序列和长期行为序列。模型仍然以当前候选物品作为目标，预测用户对该物品的点击概率。其中，短期行为序列和基础画像特征参与常规的特征交互与兴趣建模，而 CAIN 模型重点改造的是长期行为序列的处理过程。

传统的长序列注意力通常将历史序列中的每个行为单独与候选物品进行匹配，而 CAIN 模型首先通过 TCN 为这些行为生成包含局部上下文的表示，再将其送入后续注意力模块。这样，注意力模块不仅评估某个历史物品本身与候选物品的相关性，还能够利用该物品在连续行为中的上下文信息判断其对用户当前兴趣的意义。

假设用户的长期行为序列为 $\mathbf{LH} = \{\ell h_1, \ell h_2, \ldots, \ell h_N\}$，其中 ℓh_t 表示序列中的第 t 个历史行为。对于中心行为 ℓh_t，CAIN 模型将其前后各 c_l 个行为作为上下文，其上下文集合可以表示为：

$$\mathcal{C}_t^{(c_l)} = \{\ell h_{t-c_l}, \ldots, \ell h_{t-1}, \ell h_{t+1}, \ldots, \ell h_{t+c_l}\} \tag{20.171}$$

这里的 c_l 表示中心行为在序列前后各自考虑的上下文长度。由于卷积窗口需要同时覆盖中心行为及其相邻行为，因此对应的卷积核宽度为 $2c_l + 1$。在实际计算中，首先将窗口内各行为的 Embedding 按照序列顺序拼接，并保留中心行为本身的信息，然后通过共享的卷积参数进行线性变换，得到中心行为的上下文感知表示：

$$\mathbf{cr}_t = \left[\mathbf{e}_{\ell h_{t-c_l}}; \ldots; \mathbf{e}_{\ell h_{t-1}}; \mathbf{e}_{\ell h_t}; \mathbf{e}_{\ell h_{t+1}}; \ldots; \mathbf{e}_{\ell h_{t+c_l}}\right] \mathbf{W}^C + \mathbf{b}^C \tag{20.172}$$

其中，$\mathbf{e}_{\ell h_t}$ 表示中心行为的 Embedding，$\mathbf{W}^C$ 和 $\mathbf{b}^C$ 分别表示卷积层的权重矩阵和偏置项，$\mathbf{cr}_t$ 则是融合了中心行为及其相邻行为信息的上下文感知表示。对于序列首尾缺少足够相邻行为的位置，模型采用零填充，使各位置的卷积窗口保持一致的尺寸。

需要注意的是，这里的 TCN 采用以中心行为为基准的双向局部上下文窗口，即同时利用该行为在序列中前后相邻的历史行为。这与面向未来序列预测任务的严格因果卷积有所不同。由于推荐场景中的整个长期行为序列均来自当前预测请求之前已经发生的用户行为，因此我们可以利用序列中某个位置前后的历史信息共同理解该位置所代表的用户反馈。

在得到上下文感知表示后，CAIN 模型不再直接使用原始行为 Embedding 计算注意力，而是以 $\mathbf{cr}_t$ 替代 $\mathbf{e}_{\ell h_t}$，与候选物品表示进行相关性计算：

$$s_t = \mathrm{Attn}(\mathbf{v}_e, \mathbf{cr}_t; \theta_a) \tag{20.173}$$

其中，$\mathbf{v}_e$ 表示候选物品的 Embedding，θ_a 表示注意力模块的参数。后续的注意力权重计算、行为筛选和兴趣聚合等操作均基于上下文感知表示展开。这样，即使某个历史行为本身与候选物品具有较高的语义相关性，模型

也能够结合它前后发生的行为变化，重新评估该行为对当前用户兴趣的实际意义。

20.4.7.0.1 多尺度兴趣聚合。 单一卷积窗口只能捕获固定范围内的局部上下文，而不同范围的历史行为可能反映出不同层次的用户兴趣变化。较短的上下文通常更关注某个行为发生后用户的即时反馈，例如用户是否继续浏览相似内容；较长的上下文则能够包含更广泛的行为变化，有助于识别用户兴趣的迁移或衰减。为了同时利用这些信息，CAIN 模型设计了 Multi-Scope Interest Aggregator（MSIA）模块，通过堆叠多个 TCN 层逐步扩大感受野，并在不同层的输出上分别执行目标感知的注意力计算。

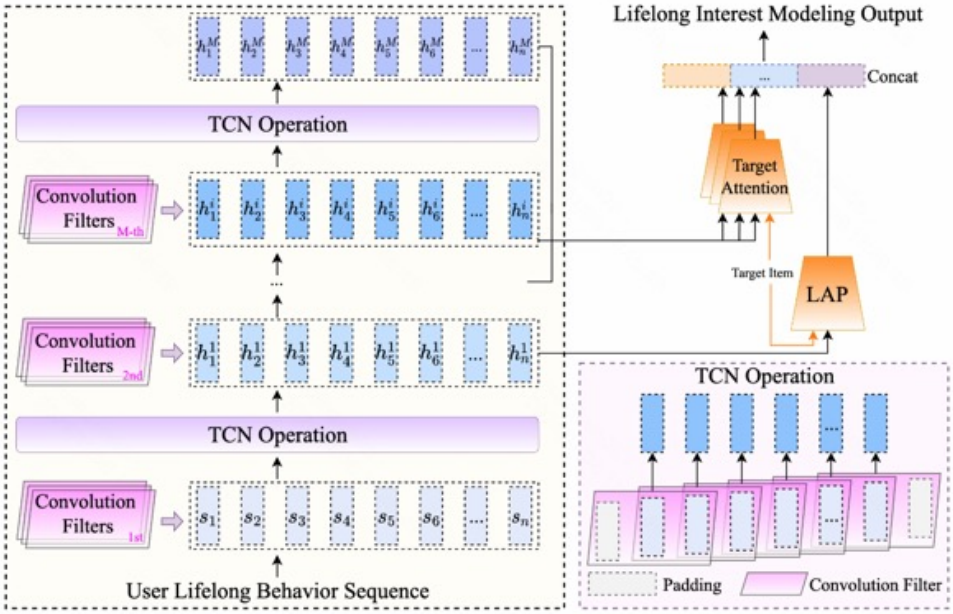


图 20.31: MSIA 模块的网络结构。

如图20.31所示，MSIA 模块由多个串联的 TCN 层及其对应的注意力模块组成。随着网络层数增加，每一层的输出都能够融合更大范围内的历史行为信息，从而形成不同上下文范围的行为表示。假设第 n 层的上下文长度为 $c_l^{(n)}$，当卷积步长为 1 时，下一层的上下文长度可以表示为：

$$c_l^{(n+1)} = c_l^{(n)} + f_s^{(n+1)} - 1 \tag{20.174}$$

其中，$f_s^{(n+1)}$ 表示第 $n+1$ 层的卷积核尺寸。通过逐层堆叠卷积，后续层的感受野不断扩大，其输出表示能够覆盖更长范围内的行为上下文。与单独调整一层 TCN 的卷积核尺寸相比，堆叠结构不仅能够产生多个不同范围的上下文表示，还能够利用网络中的非线性变换进一步提取高层行为特征。

在长序列场景中，如果所有 TCN 层都保持与原始序列相同的长度，那么后续注意力模块仍然需要处理大量位置，难以充分发挥上下文卷积的效率优势。因此，CAIN 模型将第一层 TCN 的卷积步长固定为 1，以保证长期行为序列中的每个原始行为都拥有独立的上下文感知表示；对于后续 TCN 层，则采用大于 1 的卷积步长，使多个相邻位置的信息逐步合并到更少的输出位置中。这样，随着网络加深，输出序列长度不断缩短，后续注意力模块需要处理的行为表示数量也相应减少。同时，较大的步长能够加快感受野的扩展，使模型以较少的卷积层覆盖更长范围的上下文。由于深层表示关注的是较大范围内的行为模式，单个原始行为是否仍然保留独立表示就不再是主要目标。

由于不同 TCN 层的输出具有不同的序列长度和上下文范围，因此，MSIA 模块为各层配置独立的注意力模块，避免不同层之间共享注意力参数所带来的相互影响，也便于不同层级的兴趣提取并行执行。其中，第一层 TCN 的输出长度与原始长期行为序列一致，因此仍然采用 LAP 作为主要注意力模块，提取细粒度的长序列兴趣表示：

$$\mathbf{IR}_1 = \mathrm{LAP}(\mathbf{v}_e, \mathbf{CR}^{(1)}) \tag{20.175}$$

这里的 $\mathbf{CR}^{(1)}$ 表示第一层 TCN 输出的上下文感知行为序列。沿用 LAP 作为主注意力模块，使得 CAIN 模型能

够保留原有长序列建模方法在大规模历史行为筛选方面的能力，同时将输入替换为包含相邻行为信息的上下文表示。

对于后续的 TCN 层，由于其输出序列已经通过步长卷积进行了压缩，待处理的位置数量明显少于原始长期行为序列，因此可以采用结构更简单的目标注意力。第 n 层的辅助注意力首先分别通过线性映射得到候选物品的 Query，以及上下文感知行为表示的 Key 和 Value，再利用缩放点积计算候选物品与各位置之间的相关性：

$$s_{a,t}^{(n)} = \frac{(\mathbf{W}_n^Q \mathbf{v}_e)^\top (\mathbf{W}_n^K \mathbf{cr}_t^{(n)})}{\sqrt{d}} \tag{20.176}$$

其中，$\mathbf{W}_n^Q$、$\mathbf{W}_n^K$ 和 $\mathbf{W}_n^V$ 分别表示第 n 层辅助注意力中的 Query、Key 和 Value 映射矩阵，d 表示映射后的特征维度，$\mathbf{cr}_t^{(n)}$ 表示该层第 t 个位置的上下文感知表示。根据上述相关性得分，对各位置的 Value 进行加权聚合，得到第 n 层对应的兴趣表示：

$$\mathbf{IR}_n = \sum_{t=1}^{T_n} s_{a,t}^{(n)} \left(\mathbf{W}_n^V \mathbf{cr}_t^{(n)} \right) \tag{20.177}$$

其中，T_n 表示第 n 层 TCN 的输出序列长度。由于不同层使用独立的注意力参数，各层可以针对自身所覆盖的上下文范围学习不同的目标相关性。最后，MSIA 模块将所有层得到的兴趣表示进行拼接，形成融合多种上下文范围信息的用户兴趣表示：

$$\mathbf{IR}_{\text{MSIA}} = [\mathbf{IR}_1; \mathbf{IR}_2; \ldots; \mathbf{IR}_L] \tag{20.178}$$

其中，L 表示 TCN 层数。所以第一层的 LAP 主要负责从完整长期行为序列中提取细粒度的目标相关兴趣，后续层的辅助注意力则从更大范围的上下文表示中提取相对粗粒度的兴趣信息。两者相互补充，使最终表示既保留了具体历史行为的相关性，也能够反映更长时间范围内的用户兴趣变化。

20.4.7.0.2　个性化卷积生成。 上述 TCN 通过卷积核在用户行为序列上提取局部上下文，其卷积参数在不同用户之间共享。这种设计隐含地假设不同用户的行为模式可以由相同的上下文提取方式描述，但在实际推荐场景中，用户的活跃程度、内容偏好以及行为对后续兴趣的影响程度往往存在较大差异。例如，对于活跃用户而言，其观看决策可能更多取决于当前物品自身的内容特征，前序行为对其后续选择的影响相对有限；而对于行为模式相对稳定或更容易受到外部内容影响的用户，连续行为之间的上下文关系可能更加重要。因此，即使面对相同的行为序列模式，不同用户也可能需要不同的上下文提取方式。

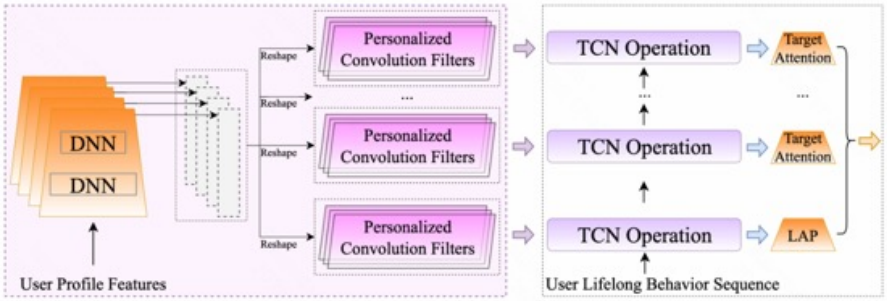


图 20.32: PEG 模块的网络结构。

因此，CAIN 模型进一步引入 Personalized Extractor Generation（PEG）模块，根据用户的基础画像特征动态生成个性化卷积核，从而让不同用户使用不同的 TCN 参数提取行为上下文。PEG 的设计受到 POSO 等个性化网络结构的启发，其核心并不是为每个用户单独维护一套完整的卷积网络，而是通过一个轻量级的全连接网络，将用户基础特征映射为卷积核参数。整体结构如图20.32所示。关于个性化网络参数方法的讨论，我们将会在本书第24章《个性化网络参数》中进行详细介绍。

假设 $\mathbf{I}$ 表示用户基础画像特征的 Embedding 表示。PEG 针对第 n 层 TCN 配置独立的两层全连接网络，首先通过第一层提取用户特征的交互表示：

$$\mathbf{O}_n = \text{ReLU}(\mathbf{W}_n^{P1} \mathbf{I} + \mathbf{b}_n^{P1}) \tag{20.179}$$

随后，第二层将其映射到对应 TCN 层的卷积参数空间，并通过 Reshape 操作得到个性化卷积核：

$$\mathbf{W}_n^{PC} = \text{Reshape}(\mathbf{W}_n^{P2}\mathbf{O}_n + \mathbf{b}_n^{P2}) \tag{20.180}$$

其中，$\mathbf{W}_n^{P1}$、$\mathbf{b}_n^{P1}$ 以及 $\mathbf{W}_n^{P2}$、$\mathbf{b}_n^{P2}$ 分别表示两层全连接网络的权重和偏置，$\mathbf{W}_n^{PC}$ 表示为当前用户生成的第 n 层个性化卷积核。PEG 的输入不仅可以包含年龄、性别、地理位置和教育背景等基础画像特征，还可以包含用户的行为统计信息，例如历史曝光量、点击量以及用户交互较多的作者等。不同类型的特征共同为卷积核生成提供用户侧条件信息。

在得到个性化卷积核后，CAIN 模型使用其替换原本在所有用户之间共享的全局卷积核，完成对应 TCN 层的上下文提取。对于第 n 层，其计算可以表示为：

$$\mathbf{cr}_t^{(n)} = \mathbf{x}_t^{(n)}\mathbf{W}_n^{PC} + \mathbf{b}_n^{C} \tag{20.181}$$

其中，$\mathbf{x}_t^{(n)}$ 表示第 n 层中以当前位置为中心的卷积窗口输入，由该层输入序列中相邻位置的表示拼接得到；$\mathbf{b}_n^{C}$ 为卷积偏置项。由于 $\mathbf{W}_n^{PC}$ 由用户画像特征动态生成，因此同一层 TCN 在处理不同用户的历史行为时，可以采用不同的卷积参数，从而学习更加符合用户行为特征的上下文表示。

CAIN 模型的实验中也比较了将全局卷积核与个性化卷积核同时使用、再融合两者输出的方式。相比于保留两套卷积结果进行融合，直接使用个性化卷积核替换全局卷积核能够获得相近且更有利于控制模型结构复杂度的效果。因此，CAIN 最终采用由 PEG 生成的个性化卷积核执行上下文提取。

总体而言，CAIN 模型是在 LAP 长序列建模基础上引入上下文感知能力的一种扩展。LCN 模型所采用的 LAP 主要解决如何从超长历史行为中逐层筛选和聚合目标相关信息，而 CAIN 模型进一步关注每个历史行为与其相邻行为之间的联系，通过 TCN 构建上下文感知表示，并利用 MSIA 从多个上下文范围提取目标相关兴趣。PEG 则进一步考虑不同用户的行为模式差异，使卷积核能够根据用户特征动态调整。需要指出的是，CAIN 模型并没有取代 LAP，而是将其保留为第一层 TCN 输出上的主要注意力模块。这种上下文建模方式并不依赖特定的长序列注意力结构，原则上可以与 ETA、SDIM 等其他长序列建模方法结合。这样长序列建模不再局限于判断单个历史行为与候选物品之间的相关性，而是进一步利用行为发生时的局部上下文，以及不同时间范围内的行为变化，提升用户兴趣表示的完整性。

20.5 基于哈希的用户行为序列建模

随着用户历史行为序列不断增长，传统 Transformer 中的全量 Attention 会带来较高的计算和存储开销。对于长度为 L 的用户行为序列，标准自注意力机制需要计算任意两个行为之间的相关性，其注意力矩阵的规模为 $L \times L$，计算复杂度和存储复杂度均随序列长度呈平方级增长。在工业级推荐系统中，用户可能积累数千甚至数万条历史交互记录，如果直接使用全量 Attention 对这些行为进行建模，不仅会增加模型训练和推理的资源开销，也会限制可利用的历史行为规模。

为了降低长序列建模的计算成本，一类方法尝试利用哈希技术筛选与当前行为更加相关的历史行为，从而减少不必要的两两交互。其基本思想如下：

定义 20.6

基于哈希的用户行为序列建模方法将表示空间中相似的向量映射到相同的哈希桶中，使每个位置只需要与少量同桶或相邻分块中的位置计算注意力，而不必遍历整个序列。与传统的固定窗口稀疏注意力不同，基于哈希的方法能够根据行为表示之间的相似程度动态建立关联，因此有机会在降低计算开销的同时保留较为重要的长距离依赖关系。

本节首先介绍 Reformer 模型，分析其提出的局部敏感哈希注意力机制，以及可逆 Transformer 和分块计算等配套设计。然后将讨论受到 Reformer 启发的用户行为序列建模方法，包括 ETA、SDIM 等模型，以及它们在工业推荐系统中的应用。

20.5.1 Reformer 模型

Reformer 模型源自于谷歌发表在 ICLR 2020 会议的论文《Reformer: The Efficient Transformer》[15]。该工作主要针对 Transformer 在长序列建模过程中存在的计算和内存开销问题展开研究，通过局部敏感哈希（Locality-Sensitive Hashing，LSH）重新设计注意力计算，并结合可逆残差结构和分块计算进一步降低模型的显存占用。与传统 Transformer 相比，Reformer 在保留多头注意力机制基本结构的基础上，将全量注意力计算转换为基于哈希桶的稀疏注意力计算，使其能够更加高效地处理超长序列。在工业级推荐系统中，Reformer 模型并不是一个直接面向 CTR 预测任务提出的模型，但其通过哈希筛选相似行为、减少长序列两两交互的设计，为大规模用户行为序列建模提供了一种具有代表性的技术思路。

20.5.1.0.1 局部敏感哈希注意力 首先来看 Transformer 中标准的缩放点积注意力（Scaled Dot-Product Attention）。对于输入的 Query、Key 和 Value 矩阵 Q、K 和 V，注意力计算过程可以表示为：

$$\text{Attention}(Q,K,V)=\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{20.182}$$

其中，d_k 表示 Key 和 Query 的向量维度。对于长度为 L 的序列，QK^T 会产生一个 $L\times L$ 的注意力矩阵，其中每个元素表示两个序列位置之间的相关性。标准 Transformer 通过多头注意力机制，从不同的表示子空间中学习序列位置之间的关联，但每个注意力头仍然需要计算所有 Query 与 Key 之间的点积。

对于较长的序列，全量注意力矩阵会占用大量显存。例如，当序列长度达到 64K 时，单个注意力头的注意力矩阵就包含约 42 亿个元素，仅使用 FP32 存储就需要约 16GB 显存。因此，如何在尽可能保留有效注意力关联的同时减少需要计算的 Query-Key 对，是长序列 Transformer 需要解决的核心问题之一。

Reformer 模型的一个关键观察是：

笔记 在 Softmax 注意力中，真正对输出产生较大影响的通常是相似度较高的少量 Key，而大量相似度较低的 Key 对最终结果的贡献相对有限。因此，没有必要让每个 Query 都与整个序列中的所有 Key 计算注意力，而是可以优先寻找与当前 Query 相似的 Key，仅在这些候选位置之间进行注意力计算。

为了高效地找到相似向量，Reformer 引入了局部敏感哈希机制。局部敏感哈希是一类用于近似高维向量相似性检索的技术，在本书第18章中有详细介绍。与普通哈希函数希望尽可能将不同输入映射到不同位置不同，局部敏感哈希的设计目标是让相似向量以较高概率被映射到同一个哈希桶，而距离较远的向量则更有可能被分配到不同的哈希桶。这样就可以利用哈希桶缩小相似向量的检索范围，避免在整个向量集合中执行代价较高的相似度计算。

Reformer 模型使用基于随机投影的角度局部敏感哈希（Angular LSH）实现这一过程。模型首先生成一个随机投影矩阵 R，其维度为 $d_k\times b/2$，其中 b 表示哈希桶的数量。对于一个输入向量 x，将其分别与 R 和 $-R$ 相乘，再将得到的两个向量拼接，通过取得最大投影值所在的位置确定哈希桶编号：

$$h(x)=\text{argmax}\left([xR;-xR]\right) \tag{20.183}$$

其中，$[xR;-xR]$ 表示两个投影结果的拼接，最终得到的哈希编号范围为 1 到 b。由于该方法根据向量在随机投影方向上的相对大小分配哈希桶，方向相近的向量更有可能获得相同的哈希结果。因此，原本需要在高维空间中进行的相似向量搜索，可以近似转换为哈希桶内的局部检索。

图20.33展示了角度局部敏感哈希的基本原理。图中通过不同的随机投影方向对向量空间进行划分，夹角较小的两个向量更有可能落入相同的哈希桶。需要注意的是，局部敏感哈希并不能保证所有相似向量都被分配到同一个桶中，其本质上是一种基于概率的近似检索方法。

在标准 Transformer 中，Query 和 Key 通常由输入表示经过两个不同的线性变换得到，两者处于不同的投影空间。而 Reformer 为了让 Query 和 Key 能够使用相同的哈希机制，采用了共享 Query-Key 投影（Shared-QK）的设计。假设输入表示为 A，则 Query 和 Key 使用同一个线性变换生成，而 Value 使用独立的线性变换：

$$Q=AW_Q,\quad K=Q,\quad V=AW_V \tag{20.184}$$

其中，W_Q 和 W_V 分别表示 Query-Key 与 Value 的投影矩阵。进一步地，Reformer 对 Key 向量进行归一化得到

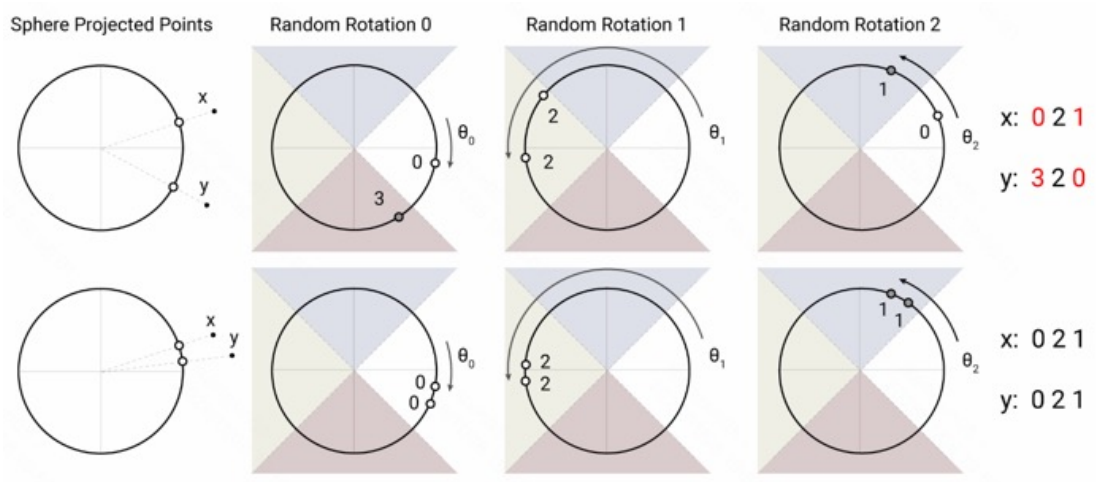


图 20.33: Reformer 中的角度局部敏感哈希示意图。

$k_j = q_j/\|q_j\|_2$。由于向量归一化不会改变其方向，因此不会改变基于角度的哈希结果。通过共享 Query-Key 投影，Query 和 Key 可以在统一的向量空间中进行哈希映射，从而便于利用哈希桶建立注意力关联。

在获得哈希函数之后，Reformer 模型将传统的全量注意力计算限制在同一个哈希桶内。对于序列中位置 i 对应的 Query q_i，Reformer 模型只允许其与哈希结果相同的 Key 计算注意力，其可访问的位置集合定义为：

$$\mathcal{P}_i = \{j \mid h(q_i) = h(k_j)\} \tag{20.185}$$

相应地，位置 i 的注意力输出可以表示为：

$$o_i = \sum_{j \in \mathcal{P}_i} \frac{\exp(q_i \cdot k_j)}{\sum_{t \in \mathcal{P}_i} \exp(q_i \cdot k_t)} v_j \tag{20.186}$$

与标准 Attention 不同，这里的 Softmax 仅在与当前 Query 位于同一个哈希桶的 Key 上进行归一化，而不再需要考虑整个序列。这样就将原本的全序列注意力计算转换为同一哈希桶内的局部注意力计算。

不过，直接按照哈希桶计算注意力还存在一些工程上的问题。一方面，不同哈希桶中的向量数量可能存在较大差异，导致各个桶的计算规模不一致，不利于 GPU 上的批量并行计算；另一方面，Query 和 Key 如果独立进行哈希分配，还可能出现同一个桶中包含较多 Query、但缺少对应 Key 的情况。

为了解决这些问题，Reformer 模型在实际计算中引入了排序与分块（Sorting and Chunking）机制。模型首先根据哈希桶编号对序列中的 Query 和 Key 进行排序，并在同一哈希桶内按照原始序列位置进一步排序。经过排序后，具有相同哈希结果的位置会聚集在相邻区域，使得原本分散在整个注意力矩阵中的有效交互关系集中到矩阵的局部区域。

随后，模型将排序后的序列按照固定大小划分为若干连续分块，每个分块中的 Query 只与当前分块以及相邻分块中的 Key 计算注意力。如果模型采用的因果注意力设置，每个位置只允许访问当前分块和前一个分块中的位置。其候选注意力集合可以表示为：

$$\widetilde{\mathcal{P}}_i = \left\{ j \mid \left\lfloor \frac{s_i}{m} \right\rfloor - 1 \le \left\lfloor \frac{s_j}{m} \right\rfloor \le \left\lfloor \frac{s_i}{m} \right\rfloor \right\} \tag{20.187}$$

其中，s_i 表示原始序列位置 i 经过排序后的新位置，m 表示每个分块包含的位置数量。由于排序后同一哈希桶中的位置集中分布在相邻区域，因此当前分块和前一个分块通常就能够覆盖大部分有效注意力关联。在实际计算时，模型仍然需要根据哈希桶编号以及因果掩码过滤掉不满足条件的位置，避免将不同哈希桶中的元素或未来位置错误地纳入注意力计算。

图20.34展示了 LSH Attention 从哈希分桶到排序、分块以及最终注意力计算的主要过程。通过上述设计，Reformer 模型不再显式构造完整的全量注意力矩阵，而是将计算集中到排序后相邻的局部区域，从而降低了长序列注意力计算的开销。需要指出的是，单次哈希仍然存在一定的近似误差。如果两个向量在实际空间中比较相似，但由于随机投影的影响被分配到了不同的哈希桶，那么它们之间的注意力关联就可能被遗漏。

为了降低这种情况发生的概率，Reformer 模型进一步引入了多轮哈希（Multi-round LSH Attention）。模型使

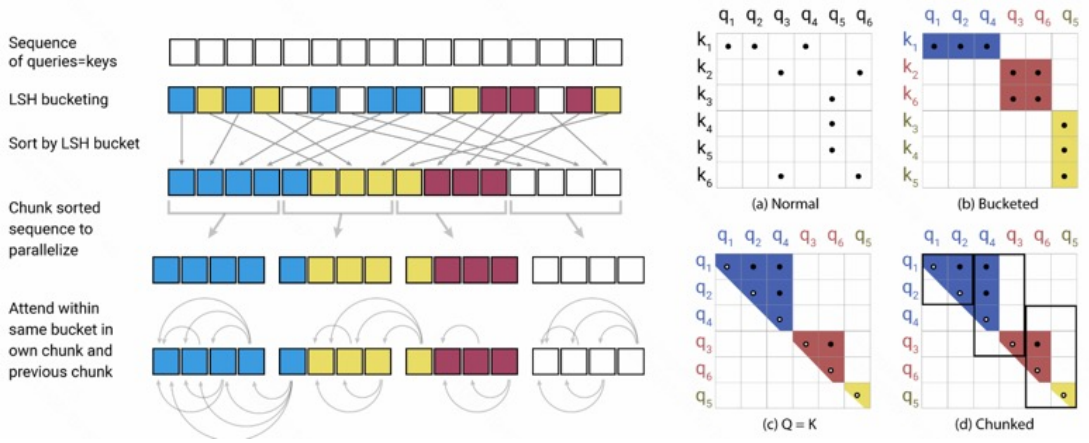


图 20.34: Reformer 中 LSH Attention 的计算过程。

用多个相互独立的随机投影矩阵生成不同的哈希函数，每一轮哈希都会产生一组候选注意力位置，最终将不同轮次得到的候选位置取并集：

$$\mathcal{P}_i = \bigcup_{r=1}^{n_{\text{rounds}}} \mathcal{P}_i^{(r)} \tag{20.188}$$

其中，n_{rounds} 表示哈希轮数，$\mathcal{P}_i^{(r)}$ 表示第 r 轮哈希为位置 i 找到的候选位置集合。通过多轮哈希，同一个 Query 可以从不同的随机投影角度寻找相似 Key，从而提高有效注意力关联被召回的概率。不过，增加哈希轮数也意味着需要执行更多次哈希、排序和局部注意力计算，因此需要在计算开销与近似精度之间进行权衡。此外，对于同时出现在多个哈希轮次中的位置，实际计算还需要对重复项进行处理，避免重复计入其注意力贡献。

从计算复杂度来看，标准 Attention 需要计算长度为 L 的序列中所有位置对之间的相关性，其主要计算和注意力矩阵存储开销均为 $O(L^2)$。而在 LSH Attention 中，通过哈希将序列划分为多个局部区域，再利用分块机制限制每个位置参与计算的候选范围。在分块大小和哈希轮数相对固定的情况下，局部注意力计算可以随序列长度近似线性增长。考虑到排序操作本身的开销，则整体的计算复杂度通常可以表示为 $O(L \log L)$，注意力矩阵的存储开销也可以由原来的平方级降低至近似线性级。因此，Reformer 模型能够在较低的计算和显存开销下处理更长的输入序列。

20.5.1.0.2　可逆 Transformer 与分块计算　除了通过 LSH Attention 降低注意力计算的复杂度之外，Reformer 模型还针对 Transformer 在深层网络训练过程中的激活值存储问题提出了可逆 Transformer（Reversible Transformer）和分块计算（Chunking）机制。

在普通 Transformer 中，每一层都会产生新的中间激活值。为了在反向传播过程中计算梯度，通常需要保存各层的中间结果。假设模型具有 N_l 层，每层的输入表示长度为 L，维度为 d_{model}，那么仅保存各层的输入激活值就需要与 $N_l L d_{\text{model}}$ 成正比的存储空间。此外，Transformer 中的前馈网络（Feed-Forward Network，FFN）通常会先将隐藏维度扩展至 d_{ff}，再经过非线性变换映射回原始维度，其中间激活值的存储开销可能进一步增加。

为了解决这一问题，Reformer 模型借鉴了可逆残差网络（Reversible Residual Networks，RevNets）的设计思想，将每一层的输入和输出组织为两路表示，并通过特殊的残差连接方式，使得上一层的输入能够根据当前层的输出和网络参数重新计算出来。

假设输入为 X_1 和 X_2，分别通过注意力层和前馈网络进行变换，可以得到：

$$Y_1 = X_1 + \text{Attention}(X_2) \tag{20.189}$$

$$Y_2 = X_2 + \text{FeedForward}(Y_1) \tag{20.190}$$

与普通残差连接不同，这种结构中的两个输出可以通过减去对应的变换结果恢复为原始输入：

$$X_2 = Y_2 - \text{FeedForward}(Y_1) \tag{20.191}$$

$$X_1 = Y_1 - \text{Attention}(X_2) \tag{20.192}$$

因此，在反向传播过程中，模型不必始终保存每一层的全部中间激活值，而是可以从当前层的输出开始，逐层反向重建上一层的输入，再计算对应的梯度。这样便能够显著减少深层 Transformer 在训练过程中需要保存的中间激活值，使激活值的存储开销不再随网络层数线性增长。需要注意的是，可逆计算并不意味着完全不需要存储中间结果，而是通过反向重建减少跨层保存激活值的需求，通常也需要额外执行相应的前向计算。

另一方面，即使采用可逆残差结构，前馈网络中的大维度中间表示仍然可能占用较多显存。由于标准 Transformer 中的前馈网络对不同序列位置分别执行相同的变换，各个位置之间不存在直接的计算依赖，因此可以将完整序列划分为多个较小的分块，逐块执行前馈网络计算。

假设将序列划分为 c 个分块，则前馈网络可以按照分块形式分别计算：

$$Y_2^{(r)} = X_2^{(r)} + \text{FeedForward}(Y_1^{(r)}), \quad r = 1, \ldots, c \tag{20.193}$$

其中，$X_2^{(r)}$ 和 $Y_1^{(r)}$ 分别表示第 r 个分块对应的输入表示和中间表示。由于不同位置的前馈网络计算相互独立，因此分块执行不会改变最终的计算结果，只是将原本需要一次性保存的较大中间张量拆分为多个较小的计算任务。通过逐块执行前向和反向计算，可以降低单次计算所需的峰值显存。

综合来看，Reformer 模型通过三种互补的设计提升了长序列 Transformer 的计算效率：LSH Attention 主要减少不同序列位置之间的注意力计算量；可逆 Transformer 通过反向重建降低深层网络的激活值存储开销；分块计算则进一步控制前馈网络中间表示的峰值显存。这些设计分别从注意力计算、网络层间激活值存储以及单层中间张量三个角度优化了长序列建模的资源开销。

对于工业级推荐系统来说，Reformer 模型的核心启发在于利用行为表示之间的相似性筛选注意力计算范围，而不是始终对整个用户历史序列执行全量交互。不过，这种方法本质上仍然是一种近似注意力机制，哈希桶的划分和随机投影可能导致部分有价值的历史行为未能参与当前注意力计算。尤其是在推荐场景中，一些较早发生但与当前候选物品高度相关的行为，未必能够在用户表示空间中与其他相似行为保持足够接近。

因此，在实际应用中，我们需要结合哈希轮数、分块大小以及行为表示的学习效果综合权衡计算效率与兴趣信息保留程度。此外，LSH Attention 主要解决序列内部的全局交互开销，并不直接解决针对当前候选物品进行目标感知兴趣提取的问题。如何进一步结合候选物品进行高效的历史行为检索，也是后续工业推荐系统长序列建模方法重点研究的方向。

20.5.2 ETA 与 ETA-NET 模型

ETA 模型源自于阿里巴巴发表在 2021 年的论文《End-to-End User Behavior Retrieval in Click-Through Rate Prediction Model》[6]。该工作受到 Reformer 中局部敏感哈希注意力机制的启发，将哈希检索应用于推荐系统中的目标感知兴趣建模，通过对用户历史行为进行高效检索，降低长序列 Target Attention 的计算开销。

ETA-Net 模型则源自于阿里巴巴发表在 DLP-KDD 2022 会议的论文《Efficient Long Sequential User Data Modeling for Click-Through Rate Prediction》[5]。该工作在 ETA 的基础上，进一步将哈希检索机制与基础 CTR 预测模型相结合，形成完整的端到端长序列建模网络，并针对工业推荐系统中的在线推理提出了相应的工程优化方案。

这两个模型的核心思想是一致的，即：

定义 20.7

对于当前需要预测的候选物品，不必让其与用户的全部历史行为进行注意力计算，而是先通过哈希机制从长行为序列中检索出与目标物品最相关的少量行为，再对筛选后的行为执行精细化的 Target Attention。♣

与 Reformer 主要解决序列内部的全局自注意力计算不同，ETA 模型直接利用目标物品与历史行为之间的相似性进行检索，更加符合推荐系统中针对不同候选物品动态提取用户兴趣的实际需求。

20.5.2.1 标准 Target Attention 的计算瓶颈

在介绍 ETA 模型之前，首先回顾推荐系统中常用的 Target Attention 机制。与 NLP 任务中的 Self-Attention 不同，推荐系统中的 Target Attention 通常将当前候选物品作为 Query，将用户历史行为序列中的物品作为 Key 和 Value，从而根据候选物品动态提取用户的相关兴趣。DIN 模型最早系统性地采用这种方式实现目标感知的兴趣建模，后续许多基于 Attention 的推荐模型也沿用了这一设计。

假设当前目标物品的 Embedding 为 $\mathbf{E}_t \in \mathbb{R}^{1\times d}$，用户行为序列的 Embedding 矩阵为 $\mathbf{E}_s \in \mathbb{R}^{L\times d}$，其中 L 表示行为序列长度，d 表示 Embedding 维度。对于第 i 个注意力头，分别通过线性变换生成 Query、Key 和 Value：

$$\mathbf{Q}_i = \mathbf{E}_t\mathbf{W}_i^Q, \quad \mathbf{K}_i = \mathbf{E}_s\mathbf{W}_i^K, \quad \mathbf{V}_i = \mathbf{E}_s\mathbf{W}_i^V \tag{20.194}$$

随后，利用缩放点积注意力计算目标物品与历史行为之间的相关性，并对 Value 进行加权求和：

$$\text{head}_i = \text{softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^T}{\sqrt{d_k}}\right)\mathbf{V}_i \tag{20.195}$$

将多个注意力头的输出进行拼接，并通过线性变换得到最终的目标感知兴趣表示：

$$\text{TA}(\mathbf{E}_t, \mathbf{E}_s) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}^O \tag{20.196}$$

其中，h 表示注意力头的数量，d_k 表示 Key 和 Query 的维度。可以看到，Target Attention 需要计算目标物品与所有 L 个历史行为之间的相关性，其主要计算复杂度为 $O(Ld)$。

在实际的推荐系统中，一次用户请求通常需要对多个候选物品进行打分。假设候选物品数量为 N_c，则每个候选物品都需要以自身 Embedding 作为 Query，重新计算与用户历史行为之间的相关性。因此，长序列 Target Attention 的主要计算复杂度会进一步增加至 $O(N_cLd)$。当用户历史行为达到数千甚至上万条，并且每次请求需要对数百至数千个候选物品进行精排时，这种计算方式很难满足工业推荐系统严格的在线延迟要求。传统方法通常通过截断历史行为序列来控制计算成本，但这又会丢失大量有价值的长期兴趣信息。因此，ETA 模型希望解决的核心问题是：

笔记 能否在不显著增加计算开销的情况下，从超长历史行为序列中快速找到与当前候选物品最相关的行为，并在此基础上执行 Target Attention？

20.5.2.2 ETA：基于 SimHash 的目标感知行为检索

ETA 模型的一个重要观察是，Target Attention 中的 Softmax 权重通常由少量相似度较高的历史行为主导。因此，对于当前目标物品而言，并不需要对所有历史行为进行完整的注意力计算，而是可以先从中检索出最相关的 Top-K 个行为，再执行标准的多头 Target Attention。

从数学上看，假设 $\mathcal{H}_u^l$ 表示用户 u 的长期历史行为集合，$\mathcal{I}_t^K$ 表示针对目标物品 t 检索得到的 Top-K 行为集合。ETA 模型将原始行为 Embedding 矩阵 $\mathbf{E}_s$ 缩减为只包含这 K 个行为的矩阵 $\mathbf{E}'_s$，然后在该矩阵上执行 Target Attention：

$$\text{LTI}(\mathbf{E}_t, \mathbf{E}_s) = \text{TA}(\mathbf{E}_t, \mathbf{E}'_s) \tag{20.197}$$

$$\mathbf{E}'_s = \text{Stack}\left(\{\mathbf{e}_i \mid i \in \mathcal{I}_t^K\}\right) \tag{20.198}$$

其中，LTI 表示长期兴趣提取单元（Long-term Interest Extraction Unit）。整个方法的关键在于如何高效地从长度为 L 的历史行为序列中检索出这 K 个最相关的行为。

一种直观的方法是直接计算目标物品与所有历史行为 Embedding 之间的内积，再按照相似度选取 Top-K。然而，这种方法本身仍然需要 $O(Ld)$ 的计算复杂度，面对大量候选物品时并不能有效解决计算瓶颈。ETA 为此引入了 SimHash（Similarity Hashing）算法，将原本基于高维向量的相似度计算转换为基于二进制指纹的汉明距离计算。

SimHash 是一种典型的局部敏感哈希算法。其基本思想是通过多个随机投影方向将输入向量映射为二进制编码，使得原始空间中方向相近的向量更有可能得到相似的哈希结果。假设输入 Embedding 为 $\mathbf{e} \in \mathbb{R}^{1\times d}$，首先生成一个随机哈希矩阵 $\mathbf{H} \in \mathbb{R}^{d\times m}$，其中 m 表示哈希指纹的比特数，$\mathbf{H}$ 的每一列对应一个独立的随机投影方向。

SimHash 通过计算输入向量与随机投影方向之间的内积，并根据结果的正负生成二进制指纹：

$$h(\mathbf{e}) = \mathbb{I}(\mathbf{e}\mathbf{H} \geq 0) \tag{20.199}$$

其中，$\mathbb{I}(\cdot)$ 表示逐元素执行二值化操作，当投影结果大于或等于零时取值为 1，否则取值为 0。因此，一个 d 维的 Embedding 最终可以被映射为一个 m 位的二进制指纹。

对于两个输入向量 $\mathbf{e}_i$ 和 $\mathbf{e}_t$，SimHash 具有局部敏感性：如果两个向量在原始空间中的夹角较小，那么它们在随机投影后生成不同哈希位的概率也较低。对于随机超平面投影，单个哈希位发生不一致的概率与两个向量之间的夹角 θ 满足如下关系：

$$P(h_j(\mathbf{e}_i) \neq h_j(\mathbf{e}_t)) = \frac{\theta(\mathbf{e}_i, \mathbf{e}_t)}{\pi} \tag{20.200}$$

因此，两个向量在原始空间中的角度相似性，可以近似通过其二进制指纹之间的汉明距离进行衡量。汉明距离表示两个二进制编码中对应位置取值不同的比特数量，其计算方式为：

$$d_H(h(\mathbf{e}_i), h(\mathbf{e}_t)) = \text{popcount}(h(\mathbf{e}_i) \oplus h(\mathbf{e}_t)) \tag{20.201}$$

其中，$\oplus$ 表示按位异或操作，$\text{popcount}(\cdot)$ 表示统计二进制结果中 1 的数量。汉明距离越小，说明两个向量的哈希指纹越相似，也就意味着它们在原始向量空间中具有更高的相似性。

ETA 模型首先利用同一个哈希函数将目标物品和用户历史行为中的所有物品映射为二进制指纹，然后计算目标物品与各个历史行为之间的汉明距离，最终选取汉明距离最小的 K 个行为：

$$\mathcal{I}_t^K = \text{TopK}_{i \in \mathcal{H}_u^l} \left(-d_H(h(\mathbf{e}_i), h(\mathbf{e}_t))\right) \tag{20.202}$$

这样原本需要对高维 Embedding 进行大量乘加运算的相似度检索，就被转换为二进制指纹之间的异或和位计数操作。由于指纹长度通常是固定的，并且可以利用机器字长进行整数编码，因此一次汉明距离计算的开销非常低。特别是在支持硬件位运算的工业计算环境中，可以通过整数异或和 Population Count 指令高效完成相似度比较。

需要注意的是，ETA 模型采用的 SimHash 与前面介绍的 Reformer 模型虽然都属于局部敏感哈希技术，但具体应用方式有所不同。Reformer 模型主要通过多值哈希将 Query 和 Key 映射到相应的哈希桶中，再通过排序和分块完成序列内部的稀疏注意力计算，从而降低长序列自注意力的计算复杂度。而 ETA 模型则直接使用二进制指纹表示目标物品和历史行为，利用汉明距离进行目标感知的 Top-K 检索。由于推荐系统中的目标物品与历史行为并不是同一个输入，ETA 模型不要求 Query 和 Key 共享相同的投影表示，也不需要对每个用户的行为序列进行哈希排序，因而更加适合在线推荐场景。

此外，单次哈希仍然可能将原本相似的两个向量映射到不同的二进制编码中，导致部分有价值的历史行为无法被检索。为了降低这种情况出现的概率，ETA 还采用多轮哈希机制，使用不同的随机投影矩阵生成多组指纹，并综合不同轮次的哈希结果进行相似行为检索。假设哈希轮数为 n_r，第 r 轮哈希函数记为 $h^{(r)}$，则不同轮次可以分别产生一组匹配的候选行为集合：

$$\mathcal{B}^{(r)} = \{i \mid h^{(r)}(\mathbf{e}_i) = h^{(r)}(\mathbf{e}_t)\} \tag{20.203}$$

$$\mathcal{B}_t = \bigcup_{r=1}^{n_r} \mathcal{B}^{(r)} \tag{20.204}$$

通过合并多轮哈希得到的候选集合，可以提高相似历史行为被检索到的概率，再结合汉明距离完成 Top-K 筛选。多轮哈希能够在一定程度上提高检索的召回能力，但也会增加指纹存储和相似度计算的开销，因此实际应用中需要根据行为序列长度和在线延迟要求选择合适的哈希轮数。

图20.35展示了 ETA 模型的整体结构。模型将用户历史行为划分为长期行为序列和短期行为序列，两部分采用不同的兴趣提取方式。其中，长期行为序列首先经过 Embedding 层和 SimHash 检索模块，筛选出与目标物品最相关的 Top-K 个历史行为，再通过多头 Target Attention 提取长期兴趣。短期行为序列则直接通过多头 Target Attention 建模。同时，用户画像、目标物品及上下文特征也经过 Embedding 层进行编码。最后，模型将各个分支得到的兴趣表示和其他特征进行拼接，并输入 MLP 网络，经过 Sigmoid 函数输出最终的点击概率。

ETA 模型的另一个重要特点是实现了行为检索与 CTR 预测模型的端到端联合训练。传统的两阶段长序列建

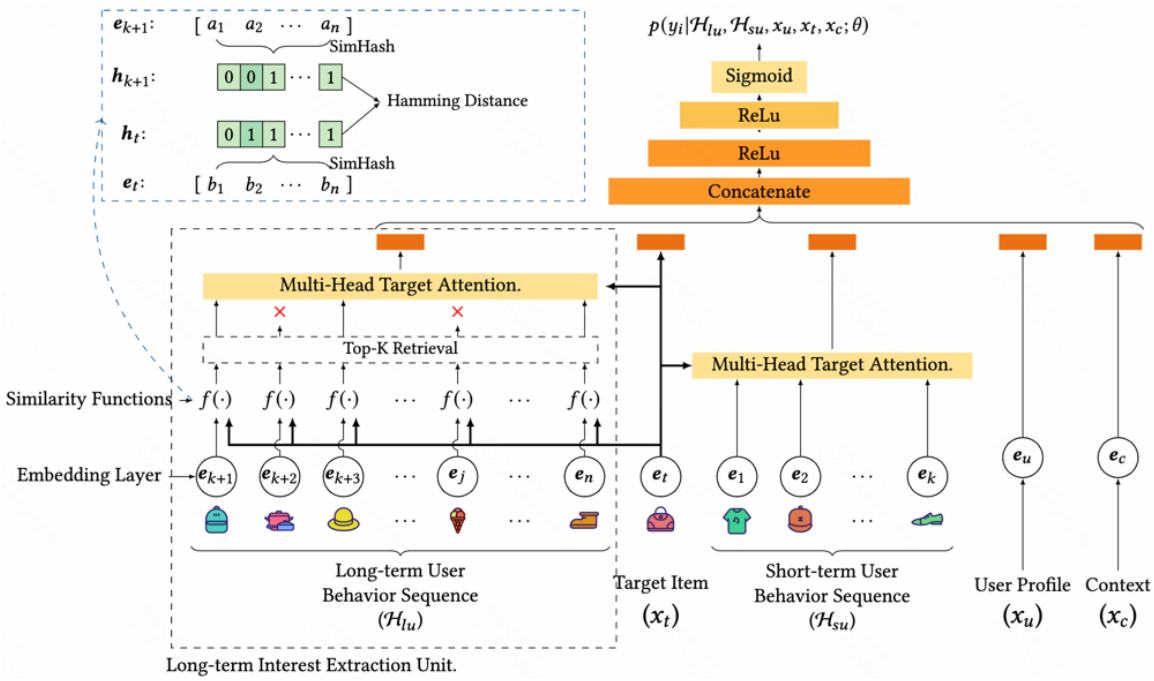


图 20.35: ETA 模型的整体结构。

模方法通常需要单独训练一个检索模型，并利用检索模型生成的 Embedding 或其他属性建立离线索引。当 CTR 模型持续更新时，检索模型使用的向量表示可能与 CTR 模型中的最新 Embedding 不一致，进而造成检索目标与最终预测目标之间的偏差。

ETA 模型则直接使用 CTR 模型 Embedding 层输出的用户和物品表示生成 SimHash 指纹。其哈希矩阵由随机矩阵初始化后固定，不需要额外训练哈希参数，也不需要通过反向传播更新哈希函数。当 Embedding 随着 CTR 模型训练不断更新时，对应的二进制指纹也会根据最新 Embedding 重新计算，从而使检索结果始终与当前 CTR 模型的表示空间保持一致。

这里需要注意的是，SimHash、汉明距离及 Top-K 检索本身属于离散操作，并不需要通过梯度下降学习检索函数。模型的梯度主要通过筛选后的原始 Embedding 和 Target Attention 部分进行传播，再由 CTR 预测损失联合更新 Embedding 层及其他网络参数。也就是说，ETA 模型实现的是检索机制与 CTR 模型的联合优化，而不是要求离散哈希和 Top-K 操作本身可微分。

从计算复杂度来看，假设每个用户请求需要评估 N_c 个候选物品，历史行为序列长度为 L，Embedding 维度为 d，检索得到的行为数量为 K。传统 Target Attention 的主要计算复杂度为 $O(N_cLd)$。采用了 ETA 模型之后，每个候选物品首先需要与长度为 L 的历史行为指纹进行汉明距离比较，再对筛选出的 K 个行为执行多头 Target Attention，其主要计算复杂度可以降低为 $O(N_c(L+Kd))$，$K \ll L$，其中 $O(N_cL)$ 对应基于汉明距离的历史行为检索，$O(N_cKd)$ 对应检索结果上的 Target Attention。相较于原始的 $O(N_cLd)$ 复杂度，该方法将高维向量相似度计算替换为低成本的整数位运算，使得模型能够在保留更长历史行为序列的同时控制在线计算开销。

20.5.2.3 ETA-Net：面向工业部署的长序列建模网络

在 ETA 模型提出之后，阿里巴巴进一步发表了 ETA-Net 模型，将上述高效 Target Attention 机制整合到完整的 CTR 预测网络中。与 ETA 模型主要介绍长序列行为检索和兴趣提取方法不同，ETA-Net 模型进一步关注如何在工业推荐系统中实现端到端训练与高效在线服务，使基于哈希的长序列建模能够真正应用于大规模推荐场景。

ETA-Net 模型整体采用 Embedding 与 MLP 相结合的基础 CTR 模型，并在此基础上增加一个专门处理长期行为序列的 ETA 分支。具体而言，用户画像、上下文特征以及目标物品特征通过 Embedding 层进行编码，短期行为序列则采用标准多头 Target Attention 提取目标感知兴趣。对于长期行为序列，模型首先通过 SimHash 生成目标物品和历史行为物品的二进制指纹，再利用汉明距离筛选 Top-K 相关行为，最后通过多头 Target Attention

提取长期兴趣表示。

因此，ETA-Net 模型可以看作由两个主要部分构成：一部分是负责长行为序列检索与建模的 ETA 模块，另一部分是负责其他用户、物品、上下文及短期行为特征建模的基础 CTR 网络。两个部分的输出在拼接后共同输入 MLP，最终得到目标物品的 CTR 预测结果。图20.36展示了 ETA-Net 的网络结构。与原始 ETA 模型相比，ETA-Net 模型更加突出哈希检索模块与基础 CTR 模型之间的整体协作关系：长期行为分支负责从大量历史交互中筛选与目标物品相关的行为，基础模型则继续处理常规特征及短期行为，两者共同完成用户对当前候选物品的兴趣估计。

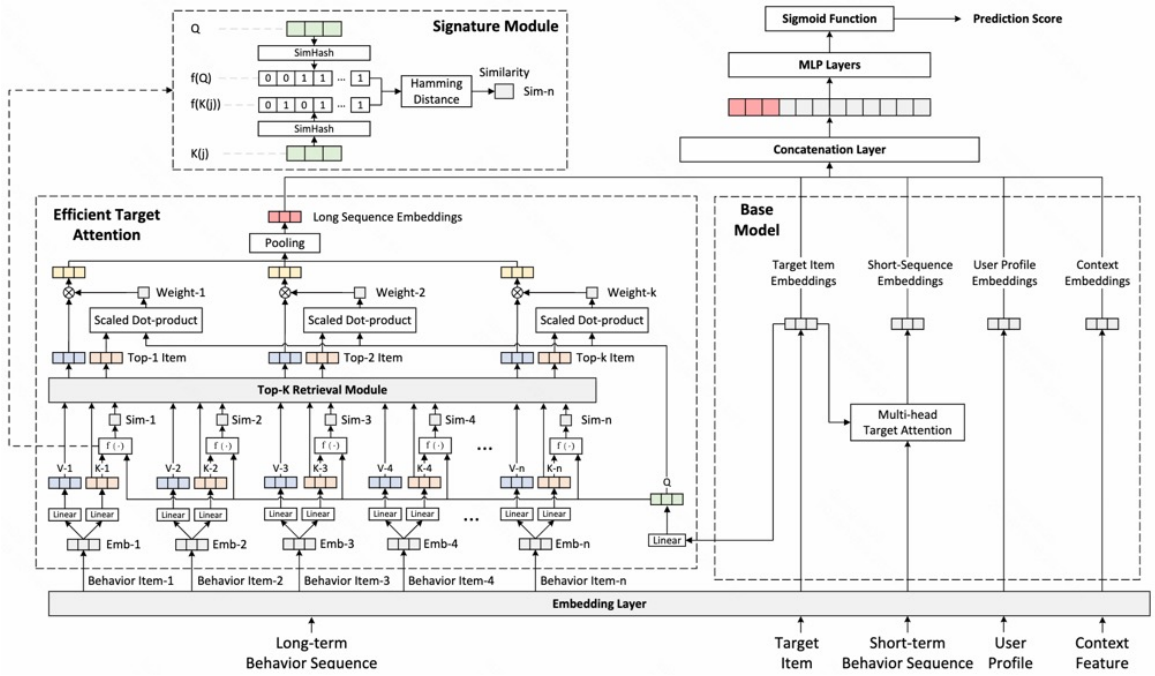


图 20.36: ETA-Net 的整体网络结构。

在训练过程中，ETA-Net 模型同样采用端到端的优化方式。哈希检索部分不包含需要通过梯度下降更新的参数，而基础 CTR 模型、Embedding 层以及 Target Attention 中的可训练参数则通过最终的 CTR 预测损失联合优化。由于目标物品和历史行为使用的是同一个 Embedding 层中的最新表示，因此训练过程中可以持续更新相应的 SimHash 指纹，避免独立检索模型造成的表示不一致问题。

在此基础上，ETA-Net 模型进一步针对线上服务进行了两项重要的工程优化。首先，预先计算并存储物品的哈希指纹。由于在线推理时模型参数通常在一个模型版本的生命周期内保持固定，而 SimHash 使用的随机投影矩阵也不需要训练，因此可以在离线模型训练完成后，遍历物品 Embedding 表，提前为所有物品生成对应的二进制指纹，并将其存储在 Embedding 查找表中。在线推理时，无论是当前目标物品还是用户历史行为中的物品，都可以直接通过物品 ID 获取预先计算好的指纹，而不必在每次请求中重复执行 SimHash 投影计算。

这种方式与传统两阶段检索方法中的离线索引有所不同。ETA-Net 模型预计算的指纹直接来源于已经训练完成的 CTR 模型 Embedding，并与该模型版本使用的 Embedding 保持一致，而不是由一个独立训练的检索模型生成。因此，它既保留了预计算和查表带来的工程效率，又不会引入独立检索模型与 CTR 模型之间的表示差异。在模型版本更新时，相应的指纹也需要同步更新，以保证线上检索使用的指纹与当前模型参数一致。

其次，ETA-Net 模型使用整数存储二进制指纹。由于 SimHash 生成的指纹本质上是由 0 和 1 组成的二进制序列，因此可以直接使用固定宽度的整数进行编码。例如，对于长度为 64 位的指纹，可以使用一个 INT64 整数进行存储。在计算两个指纹的汉明距离时，只需要执行整数异或操作，再统计结果中 1 的数量即可。这样不仅能够降低大规模物品指纹的存储开销，还可以充分利用 CPU 提供的位运算指令加速相似度比较。

通过这些优化，ETA-Net 模型可以在一次用户请求中获取用户的长期行为序列，并针对多个候选目标物品分别执行轻量级的指纹比较和 Top-K 检索。对于每个目标物品，检索过程只需要遍历历史行为指纹，而不再需

要对全部历史行为 Embedding 进行高维向量运算。随后，模型只需要对少量筛选后的行为执行完整的多头 Target Attention，从而显著降低整体在线推理成本。

从 ETA 到 ETA-Net 模型，我们可以看到基于哈希的用户行为序列建模逐渐从单一的相似行为检索算法发展为完整的工业级 CTR 预测方案。其核心思想是将昂贵的高维向量相似度计算转换为低成本的二进制指纹比较，再通过标准 Target Attention 对少量检索结果进行精细化建模。这种方式既保留了目标感知兴趣提取的能力，也避免了直接对超长历史行为执行全量 Attention 所带来的计算负担。

不过，ETA 及 ETA-Net 模型的检索过程仍然需要扫描用户历史行为中的全部哈希指纹，其检索复杂度随行为序列长度线性增长，并没有将长序列检索本身转化为次线性复杂度。此外，SimHash 主要依据 Embedding 空间中的相似性筛选行为，并不能保证检索结果与完整 Target Attention 得到的注意力权重完全一致。哈希指纹的长度、哈希轮数以及 Top-K 的取值都会影响最终的兴趣提取效果。因此，在实际落地时，工程师们需要在检索召回能力与在线计算成本之间进行权衡。

20.5.3 SDIM 模型

SDIM 模型源自于美团发表在 CIKM 2022 会议的论文《Sampling Is All You Need on Modeling Long-Term User Behaviors for CTR Prediction》[1]。该工作在 ETA 模型的基础上进一步探索了基于哈希采样的长序列兴趣建模方法。与 ETA 模型通过汉明距离检索 Top-K 相关行为、再执行 Target Attention 不同，SDIM 模型直接利用多轮哈希过程中目标物品与历史行为之间的碰撞概率估计注意力分布，并将发生碰撞的历史行为聚合为用户兴趣表示。此外，SDIM 模型还将用户行为序列的哈希编码与候选物品的兴趣提取过程进行解耦，通过独立的 Behavior Sequence Encoding（BSE）服务器降低在线 CTR 模型的计算开销。

图20.37展示了 SDIM 模型的整体架构。系统由 Behavior Sequence Encoding（BSE）服务器和 CTR 服务器组成。BSE 服务器负责对用户历史行为进行哈希编码和分桶，CTR 服务器则根据候选物品的哈希结果获取对应的行为表示，并结合其他特征完成 CTR 预测。通过这种方式，用户行为序列的哈希处理不再依赖具体的候选物品，可以在候选物品打分之前独立完成。

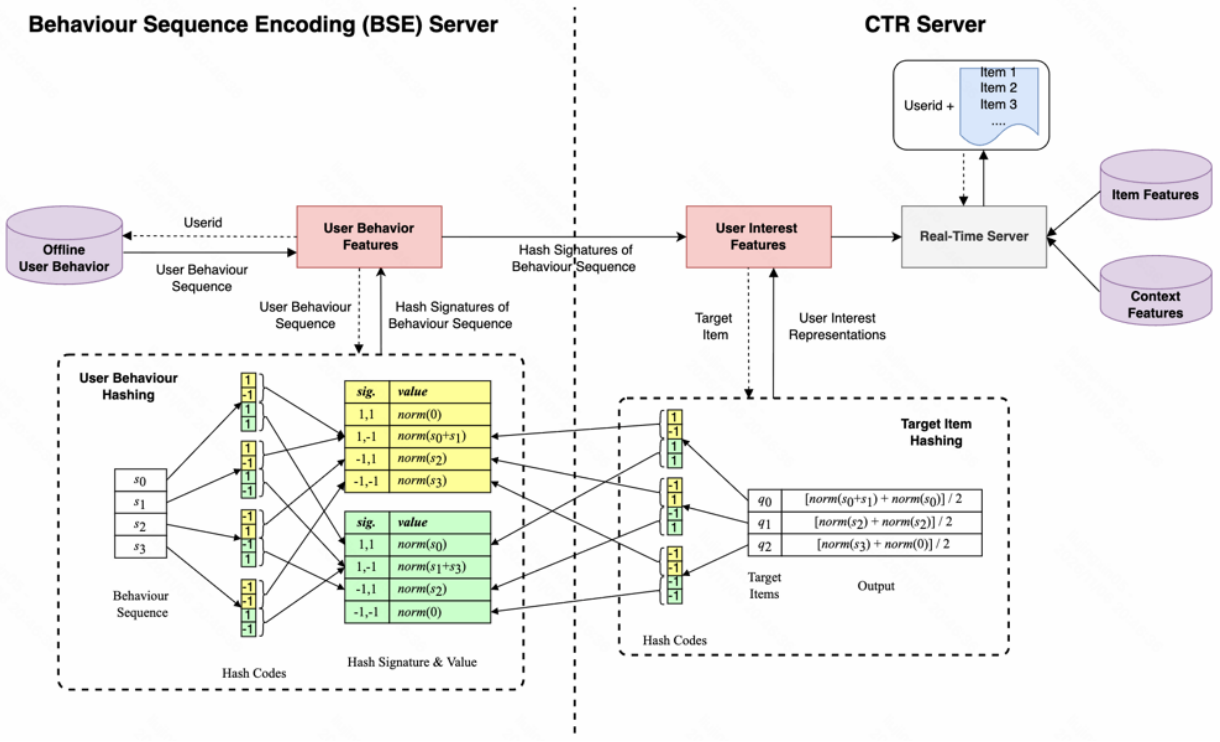


图 20.37: SDIM 的整体服务架构。

20.5.3.0.1 基于哈希采样的兴趣建模 SDIM 模型的核心观察是：

笔记 根据局部敏感哈希的性质，两个向量越相似，在随机哈希下发生碰撞的概率就越高。因此，可以通过多轮哈希的碰撞频率估计历史行为与当前目标物品之间的相关程度，而不必像 ETA 模型一样显式计算汉明距离并进行 Top-K 排序。

假设当前目标物品的 Embedding 为 $\mathbf{q} \in \mathbb{R}^d$，用户历史行为序列为 $S = \{\mathbf{s}_1, \ldots, \mathbf{s}_L\}$。对于一个随机哈希向量 $\mathbf{r}$，利用 SimHash 将输入向量映射为二进制编码：

$$h(\mathbf{x}, \mathbf{r}) = \mathbb{I}(\mathbf{r}^T \mathbf{x} \geq 0) \tag{20.205}$$

其中 $\mathbf{r}$ 的各维度从标准正态分布中独立采样。对于目标物品和历史行为，二者在当前哈希函数下的碰撞标记为：

$$p_j^{(r)} = \mathbb{I}\left(h(\mathbf{q}, \mathbf{r}) = h(\mathbf{s}_j, \mathbf{r})\right) \tag{20.206}$$

其中，$\mathbb{I}(\cdot)$ 为指示函数，只有当两个向量具有相同的哈希编码时，$p_j^{(r)}$ 才取值为 1。SDIM 模型直接将所有发生碰撞的历史行为 Embedding 进行求和，并通过 L2 归一化得到当前哈希函数下的用户兴趣表示：

$$\text{Attn}^{(r)}(\mathbf{q}, S) = \ell_2 \left(\sum_{j=1}^{L} p_j^{(r)} \mathbf{s}_j \right) \tag{20.207}$$

为了降低单次哈希带来的随机误差，SDIM 模型进一步采用多轮哈希机制。模型使用 m 个独立的随机哈希函数，并将其划分为 m/τ 组，每组包含 τ 个哈希函数。只有当目标物品与某个历史行为在同一组中的全部 τ 个哈希函数下均发生碰撞时，才认为该行为通过当前组的哈希采样：

$$\widetilde{p}_j^{(g)} = \prod_{k=1}^{\tau} \mathbb{I}\left(h(\mathbf{q}, \mathbf{r}_{g,k}) = h(\mathbf{s}_j, \mathbf{r}_{g,k})\right) \tag{20.208}$$

最终，模型分别对每组哈希采样得到的行为进行聚合和 L2 归一化，再对多组结果取平均，得到最终的长期兴趣表示：

$$\text{Attn}_{\text{SDIM}}(\mathbf{q}, S) = \frac{\tau}{m} \sum_{g=1}^{m/\tau} \ell_2 \left(\sum_{j=1}^{L} \widetilde{p}_j^{(g)} \mathbf{s}_j \right) \tag{20.209}$$

SDIM 模型的关键在于利用多轮哈希的碰撞频率近似目标感知的注意力分布。假设目标物品和历史行为 Embedding 均经过 L2 归一化，则根据 SimHash 的局部敏感性质，二者在单次随机哈希下的碰撞概率为：

$$P(h(\mathbf{q}, \mathbf{r}) = h(\mathbf{s}_j, \mathbf{r})) = 1 - \frac{\arccos(\mathbf{q}^T \mathbf{s}_j)}{\pi} \tag{20.210}$$

因此，一组包含 τ 个独立哈希函数的签名，其碰撞概率为：

$$\mathbb{E}[\widetilde{p}_j^{(g)}] = \left(1 - \frac{\arccos(\mathbf{q}^T \mathbf{s}_j)}{\pi}\right)^{\tau} \tag{20.211}$$

随着独立哈希组数量增加，实际碰撞频率可以更加稳定地逼近其期望。因此，SDIM 模型能够利用碰撞概率构造近似的注意力分布，而不需要显式计算完整的 Softmax Attention。其对应的近似注意力权重可以表示为：

$$w_j = \frac{\left(1 - \frac{\arccos(\mathbf{q}^T \mathbf{s}_j)}{\pi}\right)^{\tau}}{\sum_{l=1}^{L} \left(1 - \frac{\arccos(\mathbf{q}^T \mathbf{s}_l)}{\pi}\right)^{\tau}} \tag{20.212}$$

其中，w_j 表示历史行为 $\mathbf{s}_j$ 对应的归一化兴趣权重。这一表达式用于刻画多轮哈希下的期望注意力分布，实际模型仍然采用前面介绍的分组碰撞聚合和 L2 归一化计算用户兴趣，并不需要显式计算上述连续权重。其中，参数 τ 用于控制注意力分布的集中程度。随着 τ 增大，模型会更加关注与目标物品相似度较高的历史行为。当 τ 趋近于 0 时，碰撞概率趋近于 1，兴趣提取方式退化为 Mean Pooling。当 τ 较大时，模型则主要聚合与目标物品高度相似的行为。如果采用物品类别等离散属性进行哈希，这种方式也可以近似退化为 SIM 模型中的 Hard Search。

图20.38展示了 SDIM 模型的在线 CTR 模型结构。模型将哈希采样得到的长期兴趣、通过 Attention 提取的短期兴趣，以及用户、物品和上下文特征进行融合，最终通过 MLP 输出 CTR 的预测结果。

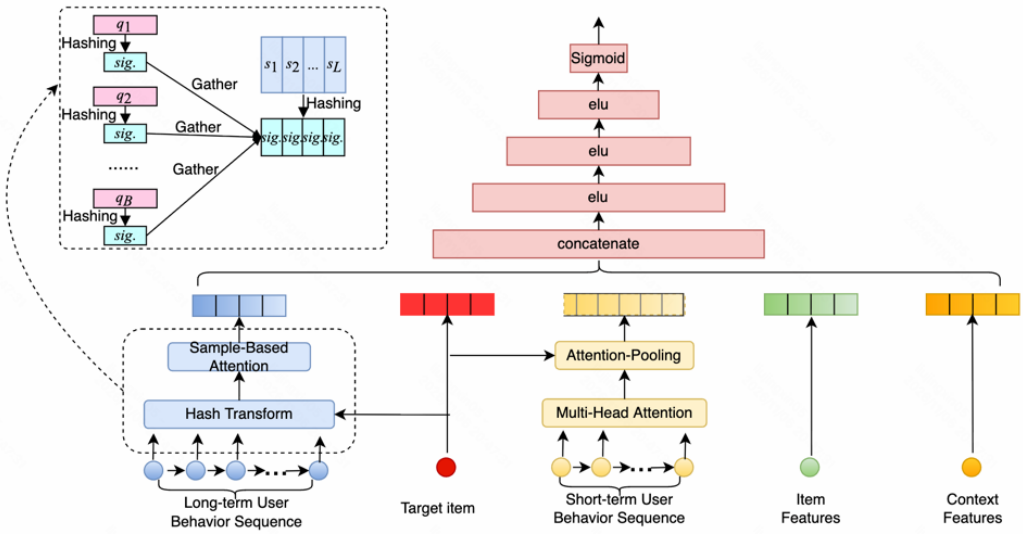


图 20.38: SDIM 在线 CTR 模型的整体结构。

20.5.3.0.2 计算优化与工业部署 SDIM 模型相比 ETA 的另一个重要改进，是充分利用了用户历史行为序列与候选物品之间的计算解耦关系。对于同一个用户请求，无论需要预测多少个候选物品，其历史行为序列都是相同的。因此，用户行为的哈希编码不需要针对每个候选物品重复计算。

假设每个请求包含 B 个候选物品，历史行为序列长度为 L，Embedding 维度为 d，哈希函数数量为 m。标准 Target Attention 的主要计算复杂度为 $O(BLd)$。SDIM 模型则将计算拆分为用户行为序列哈希和候选物品哈希两部分。借助近似随机投影算法，行为序列哈希的复杂度可以降低至 $O(Lm\log d)$，候选物品哈希的复杂度为 $O(Bm\log d)$。因此，训练阶段的主要计算复杂度为 $O(Lm\log d + Bm\log d)$。

在工业部署中，SDIM 模型进一步将这两部分拆分到 BSE 服务器和 CTR 服务器。BSE 服务器负责接收用户历史行为序列，利用固定的随机哈希函数生成行为指纹，并根据不同哈希签名对行为进行分桶和预聚合。CTR 服务器则负责对当前请求中的候选物品生成哈希编码，再根据其哈希签名从 BSE 服务器提供的用户行为哈希桶中获取对应的聚合表示，最终与其他用户、物品及上下文特征一起输入 CTR 模型完成预测。

由于哈希函数数量固定，BSE 服务器只需要向 CTR 服务器传输固定规模的哈希桶表示，而不必传输完整的历史行为序列。论文报告，其线上系统中单次请求的行为表示传输量约为 8KB，传输耗时约为 1ms。对于 CTR 服务器而言，长期兴趣提取部分的主要计算开销则降低为候选物品的哈希计算 $O(Bm\log d)$，这一开销不再直接依赖历史行为序列长度 L。此外，SDIM 模型在训练阶段仍然采用端到端联合优化方式，只有在线推理阶段才将 BSE 与 CTR 两个部分独立部署，因此无需改变原有 CTR 模型的主体结构，即可将哈希采样得到的长期兴趣表示作为新的特征接入现有网络。

总体来看，SDIM 模型在 ETA 的基础上进一步利用多轮哈希碰撞概率近似目标感知的注意力分布，省去了显式的汉明距离计算和 Top-K 检索，并通过 BSE 与 CTR 服务器的解耦避免了多个候选物品重复处理同一份用户历史行为。不过，SDIM 模型的兴趣表示仍然依赖哈希碰撞对向量相似度的近似估计，其效果受到哈希函数数量和参数 τ 的影响。同时，历史行为的哈希编码、分桶与聚合仍然需要由 BSE 服务器承担。因此，实际应用中还需要综合考虑兴趣建模效果、行为维护成本和线上服务开销。

20.6 基于多模态信息的用户行为序列建模

前面介绍的用户行为序列建模方法主要利用用户历史交互物品的 ID、属性或 Embedding 表示提取兴趣信息。然而，仅依赖协同过滤信号往往难以充分刻画物品的内容语义，尤其是在用户历史行为较少或物品交互数据稀疏的情况下，模型很难从有限的交互记录中准确识别用户的细粒度兴趣。随着多模态预训练技术的发展，物品的文本、图像和音频等内容信息可以被编码为具有丰富语义的 Embedding，为用户兴趣建模提供了不同于传统 ID

表示的信息来源。

将多模态信息引入用户行为序列建模，主要有两种思路。一种是在现有的行为检索框架中引入多模态 Embedding，利用物品的内容语义改善历史行为的检索效果；另一种是进一步将多模态表示转换为离散的语义 ID，并在用户行为序列中分别建模不同模态及其关联关系。接下来，本节会介绍多模态 SIM、PSRQ 等具有代表性的方法，分析多模态内容表示如何与用户历史行为检索和兴趣提取相结合。

20.6.1 多模态 SIM 模型

在前面介绍的 SIM 模型中，主要通过 Hard Search 和 Soft Search 两种方式从用户长期行为序列中检索与当前候选物品相关的历史行为。其中，Soft Search 通过目标物品与历史行为的 Embedding 相似度进行检索，从而筛选出与当前候选物品更加相关的历史交互记录。

在此基础上，可以进一步引入多模态预训练 Embedding 作为检索表示，将原本基于物品 ID 或其他语义表示的检索过程扩展到多模态内容空间。具体来说，利用预训练的多模态模型提取物品的内容 Embedding，并将当前候选物品的多模态 Embedding 作为 Query，将用户历史行为中各个物品的多模态 Embedding 作为 Key。随后，通过向量相似度计算，从历史行为序列中检索出与当前候选物品最相关的 Top-K 个行为：

$$\mathcal{H}_{u,t}^{K} = \mathrm{TopK}_{i_j^h \in \mathcal{H}_u} \, \mathrm{sim}(\mathbf{e}_t^{\mathrm{mm}}, \mathbf{e}_j^{\mathrm{mm}}) \tag{20.213}$$

其中，$\mathbf{e}_t^{\mathrm{mm}}$ 表示目标物品的多模态 Embedding，$\mathbf{e}_j^{\mathrm{mm}}$ 表示用户历史行为中第 j 个物品的多模态 Embedding，$\mathrm{sim}(\cdot)$ 表示向量相似度函数，$\mathcal{H}_{u,t}^{K}$ 表示检索得到的相关历史行为集合。

这种方式并不需要改变 SIM 模型原有的检索和兴趣建模框架，主要区别在于检索阶段使用的 Query 和 Key 由多模态预训练 Embedding 提供，而不再局限于传统的 ID 或其他协同过滤表示。通过引入物品的内容语义，模型可以在交互数据较少的情况下，仍然根据物品之间的内容相似性找到用户历史中具有潜在关联的行为。特别是对于冷启动物品，多模态 Embedding 可以提供额外的语义匹配能力，弥补仅依赖协同过滤信号所带来的检索信息不足。

需要注意的是，多模态 Embedding 在这里主要用于历史行为检索，检索得到的仍然是用户实际交互过的物品及其对应行为记录，后续可以继续沿用 SIM 模型原有的兴趣建模方式。因此，这种改进本质上是将多模态内容表示引入行为检索阶段，而不是对 SIM 模型的整体网络结构进行重新设计。

20.6.2 PSRQ 模型

PSRQ 模型源自于网易云音乐发表在 CIKM 2025 会议的论文《Progressive Semantic Residual Quantization for Multimodal-Joint Interest Modeling in Music Recommendation》[23]。该工作针对多模态推荐中的用户兴趣建模问题，提出了一种基于渐进式语义残差量化（Progressive Semantic Residual Quantization，PSRQ）的方法，将物品的多模态内容 Embedding 转换为离散的 Semantic ID，并在此基础上设计了 Multi-Codebook Cross-Attention（MCCA）网络，同时建模不同模态的用户兴趣及其交叉关联。

PSRQ 模型的整体方法分为两个阶段。第一阶段利用多模态内容特征生成模态专属和模态联合的 Semantic ID；第二阶段将这些 Semantic ID 映射为可训练的 Embedding，通过交叉注意力机制提取用户在不同模态上的兴趣表示。图20.39展示了该模型的整体流程。

20.6.2.0.1 渐进式语义残差量化 本小节首先介绍 PSRQ 如何将多模态内容 Embedding 转换为 Semantic ID。在传统的推荐系统中，物品的多模态内容通常由预训练模型提取为连续向量，这些向量包含丰富的内容语义，但直接将其用于推荐模型，一方面需要额外维护较大规模的内容特征，另一方面预训练 Embedding 通常是固定的，难以直接随推荐模型的训练过程进行优化。

为了将连续的内容表示转换为离散且可学习的语义表示，PSRQ 模型采用基于 K-means 的残差量化（Residual Quantization，RQ）作为基础。传统 RQ 首先利用 K-means 对原始 Embedding 进行聚类，得到第一层码本；随后

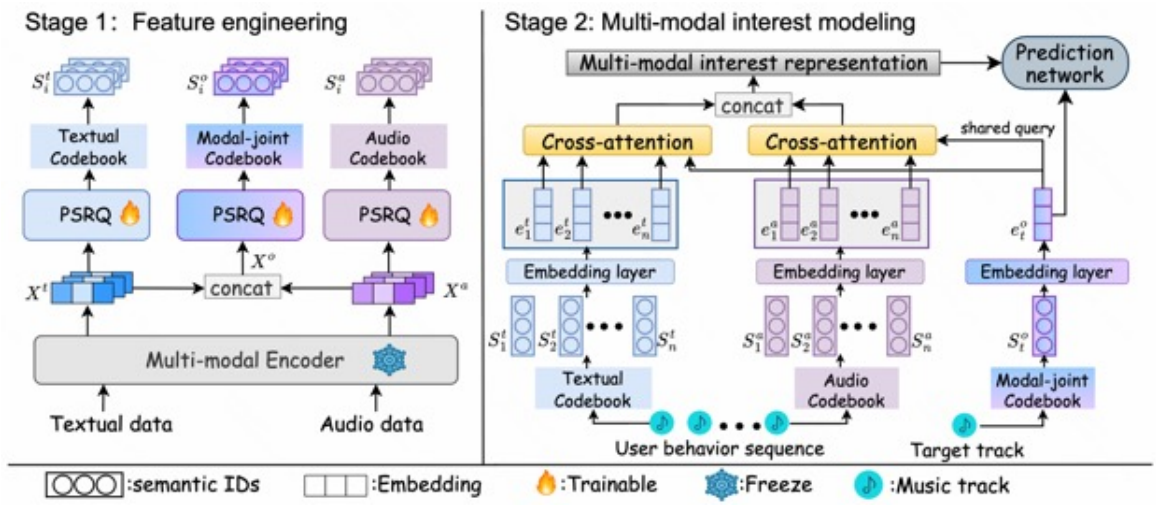


图 20.39: PSRQ 模型的整体框架。

计算原始 Embedding 与最近聚类中心之间的残差，并将残差作为下一层聚类的输入，逐层进行量化。假设原始多模态 Embedding 为 $\mathbf{X}^m \in \mathbb{R}^{|\mathcal{I}|\times d}$，其中 m 表示模态类型，d 表示 Embedding 维度，k 表示每层码本中的聚类中心数量，则传统 RQ 的过程可以表示为：

$$\mathbf{X}_1^m = \mathbf{X}^m, \quad \mathbf{C}_1^m = \mathrm{KMeans}(\mathbf{X}_1^m, k) \tag{20.214}$$

$$\mathbf{X}_l^m = \mathbf{X}_{l-1}^m - \mathrm{NearestRep}(\mathbf{X}_{l-1}^m, \mathbf{C}_{l-1}^m), \quad l = 2, \ldots, L_q \tag{20.215}$$

$$\mathbf{C}_l^m = \mathrm{KMeans}(\mathbf{X}_l^m, k), \quad l = 2, \ldots, L_q \tag{20.216}$$

其中，L_q 表示量化层数，$\mathbf{C}_l^m \in \mathbb{R}^{k\times d}$ 表示第 l 层的码本，$\mathrm{NearestRep}(\cdot)$ 表示从对应码本中检索最近的聚类中心。通过逐层量化，每一层都会进一步表示上一层尚未拟合的残差信息，最终将一个连续 Embedding 转换为由多个离散 ID 组成的 Semantic ID。

然而，传统 RQ 存在一个问题：

笔记 随着量化层数增加，后续层处理的残差向量与原始内容 Embedding 之间的语义关联可能逐渐减弱。虽然残差能够进一步刻画原始向量中尚未被表示的细节，但如果后续聚类仅依据残差进行，就可能使深层码本更加关注局部残差的数值差异，而忽略物品原本的语义信息，导致深层 Semantic ID 与原始内容之间出现语义偏移。

因此，PSRQ 对传统 RQ 进行了改进。其核心思路如下：

定义 20.8

在后续量化层中，不再仅使用当前残差向量作为聚类输入，而是将当前残差与从原始 Embedding 中减去残差后得到的前缀语义表示进行拼接，使每一层在学习剩余语义信息的同时，仍然能够保留与原始内容之间的关联。

具体来说，第一层仍然直接对原始 Embedding 进行 K-means 聚类，并计算第一层量化后的残差：

$$\mathbf{X}_1^m = \mathbf{X}^m, \quad \mathbf{C}_1^m = \mathrm{KMeans}(\mathbf{X}_1^m, k) \tag{20.217}$$

$$\mathbf{X}_2^m = \mathbf{X}^m - \mathrm{NearestRep}(\mathbf{X}_1^m, \mathbf{C}_1^m) \tag{20.218}$$

对于后续量化层，PSRQ 模型将当前残差 $\mathbf{X}_l^m$ 与原始 Embedding 中已经得到表示的部分 $\mathbf{X}^m - \mathbf{X}_l^m$ 进行拼接，构造新的聚类输入：

$$\mathbf{Z}_l^m = \mathbf{X}_l^m \oplus (\mathbf{X}^m - \mathbf{X}_l^m) \tag{20.219}$$

$$\mathbf{C}_l^m = \mathrm{KMeans}(\mathbf{Z}_l^m, k), \quad l = 2, \ldots, L_q \tag{20.220}$$

其中，$\oplus$ 表示向量拼接操作。由于第一层仅使用原始 d 维 Embedding，因此 $\mathbf{C}_1^m \in \mathbb{R}^{k\times d}$；后续层使用残差与前

缀语义的拼接表示，因此 $\mathbf{C}_l^m \in \mathbb{R}^{k\times 2d}$。通过这种方式，后续量化层在编码残差信息时，也能够保留已经由前面层表示的原始语义，从而减轻传统残差量化中深层 Semantic ID 与原始内容语义逐渐偏离的问题。

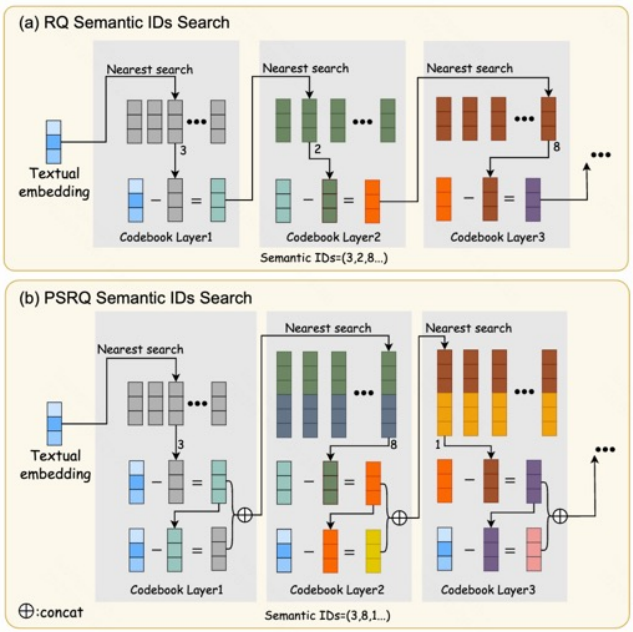


图 20.40: 传统 RQ 与 PSRQ 的 Semantic ID 生成过程对比。

图20.40对比了传统 RQ 与 PSRQ 的 Semantic ID 生成过程。传统 RQ 在每一层仅根据上一层产生的残差进行聚类，而 PSRQ 在后续层中额外引入原始 Embedding 已经得到表示的部分，使不同层的量化结果能够持续保留与原始内容之间的语义联系。在完成逐层量化后，每个物品都可以从各层码本中检索最近的聚类中心，并将其对应的聚类编号组合为 Semantic ID。假设第 l 层最近聚类中心对应的编号为 $id_{i,l}^m$，则物品 i 在模态 m 下的 Semantic ID 表示为：

$$S_i^m = [id_{i,1}^m, id_{i,2}^m, \ldots, id_{i,L_q}^m] \tag{20.221}$$

PSRQ 模型在多模态特征处理时考虑了视觉、文本和音频三种模态，但在实际线上系统中主要使用文本和音频 Embedding，即 $m \in \{t, a\}$。其中，文本 Embedding 包含歌曲名称、歌词及风格标签等语义信息，音频 Embedding 则用于表示歌曲的旋律、音色及其他声学特征。除了分别生成文本和音频的模态专属 Semantic ID 之外，PSRQ 模型还进一步构建模态联合表示。模型将文本 Embedding 和音频 Embedding 进行拼接，得到模态联合 Embedding：

$$\mathbf{X}^o = \mathbf{X}^t \oplus \mathbf{X}^a \tag{20.222}$$

其中，o 表示模态联合信息，$\mathbf{X}^o \in \mathbb{R}^{|\mathcal{I}|\times 2d}$。随后，对模态联合 Embedding 执行相同的 PSRQ 量化过程，得到对应的联合 Semantic ID S_i^o。通过分别构建模态专属码本和模态联合码本，模型既能够保留不同内容模态各自的语义信息，又能够为后续跨模态兴趣建模提供统一的语义表示。

20.6.2.0.2 多码本交叉注意力 在获得各个物品的模态专属和模态联合 Semantic ID 之后，PSRQ 模型进一步设计了 Multi-Codebook Cross-Attention（MCCA）网络，将离散语义表示融入原有的推荐模型中。其核心思想是为不同模态的 Semantic ID 分别建立可训练的 Embedding 表，并利用目标物品的模态联合表示作为共享 Query，从用户历史行为中提取不同模态的兴趣信息。

首先，在 Embedding 层中，模型并不直接使用 K-means 生成的聚类中心作为 Semantic ID 的 Embedding，而是为不同模态及不同量化层分别维护随机初始化的可训练 Embedding 表。对于 $z \in \{t, a, o\}$，第 l 层对应的 Embedding 表记为 $\mathbf{E}_l^z \in \mathbb{R}^{k\times d'}$，其中 d' 为 Semantic ID 的 Embedding 维度。对于物品 i，将其各层 Semantic ID

对应的 Embedding 进行求和，可以得到最终的模态语义表示：

$$\mathbf{e}_i^z = \sum_{l=1}^{L_q} \text{onehot}(id_{i,l}^z)\mathbf{E}_l^z \tag{20.223}$$

其中，$\text{onehot}(\cdot)$ 表示将离散的 Semantic ID 转换为 One-Hot 向量。通过这种方式，原本由多层离散 ID 组成的语义表示被映射为一个连续的可训练 Embedding。对于用户历史行为序列，可以分别得到文本、音频和模态联合的语义 Embedding 序列。

对于当前目标物品，模型仅使用模态联合码本获得其联合语义 Embedding $\mathbf{e}_t^o$，并将其作为不同模态兴趣提取时共享的 Query。这样做的目的是利用目标物品的跨模态语义信息，引导模型从不同模态的历史行为中提取与当前目标更加相关的兴趣。对于模态 $z \in \{t, a, o\}$，假设用户历史行为中第 j 个物品的对应语义 Embedding 为 $\mathbf{e}_j^z$，MCCA 网络利用目标物品的联合语义 Embedding 计算注意力权重，并对历史行为进行加权聚合：

$$\alpha_{u,j}^z = \text{Attention}(\mathbf{e}_j^z \oplus \mathbf{e}_t^o) \tag{20.224}$$

$$\mathbf{h}_u^z = \sum_{j=1}^{n} \alpha_{u,j}^z \mathbf{e}_j^z \tag{20.225}$$

其中，$\text{Attention}(\cdot)$ 由前馈网络实现，用于根据历史行为表示与目标物品联合语义表示的拼接结果生成注意力权重，$\mathbf{h}_u^z$ 表示用户在模态 z 下的兴趣表示。

与为每种模态分别使用目标物品对应的模态 Embedding 作为 Query 不同，MCCA 网络使用同一个模态联合 Embedding 作为共享 Query，使不同模态的历史行为能够在统一的目标语义条件下进行兴趣提取。例如，在音乐推荐中，用户对歌词内容的偏好与对旋律、乐器等音频属性的偏好可能存在关联，联合语义 Query 能够利用目标歌曲的整体内容信息，引导模型分别提取文本和音频兴趣，从而更好地捕获不同模态之间的互补关系。

除了多模态内容信息之外，PSRQ 模型还保留了传统协同过滤信号。模型将历史物品 ID 及目标物品 ID 分别映射为可训练的 Embedding，得到历史 ID Embedding 序列 $\{\mathbf{e}_1^r, \ldots, \mathbf{e}_n^r\}$ 和目标物品 ID Embedding$\mathbf{e}_t^r$。随后，模型采用类似的注意力机制提取用户的协同过滤兴趣表示：

$$\alpha_{u,j}^r = \text{Attention}(\mathbf{e}_j^r \oplus \mathbf{e}_t^r) \tag{20.226}$$

$$\mathbf{h}_u^r = \sum_{j=1}^{n} \alpha_{u,j}^r \mathbf{e}_j^r \tag{20.227}$$

这样，模型同时保留了不同模态的内容兴趣和基于物品 ID 的协同过滤兴趣，避免仅依赖多模态内容表示而丢失用户实际交互行为中的协同信息。

20.6.2.0.3 模型预测与优化 最后，PSRQ 模型将不同模态的用户兴趣表示与协同过滤特征进行融合，得到最终的用户兴趣表示。模型将文本、音频及模态联合兴趣表示 $\mathbf{h}_u^z$、协同过滤兴趣表示 $\mathbf{h}_u^r$，以及目标物品的协同过滤 Embedding 和模态联合语义 Embedding 进行拼接，再输入 MLP 预测用户对目标物品产生正向交互的概率：

$$\hat{y}_{u,t} = \sigma\left(\text{MLP}(\mathbf{h}_u^t \oplus \mathbf{h}_u^a \oplus \mathbf{h}_u^o \oplus \mathbf{h}_u^r \oplus \mathbf{e}_t^r \oplus \mathbf{e}_t^o)\right) \tag{20.228}$$

其中，$\sigma(\cdot)$ 表示 Sigmoid 函数。对于二分类推荐任务，模型采用交叉熵损失进行优化：

$$\mathcal{L} = -\frac{1}{N}\sum_{j=1}^{N}[y_j \log \hat{y}_j + (1 - y_j)\log(1 - \hat{y}_j)] \tag{20.229}$$

其中，N 表示训练样本数量，$y_j \in \{0, 1\}$ 表示第 j 个样本的正负反馈标签。

需要注意的是，PSRQ 模型生成 Semantic ID 的过程属于前置特征工程阶段，而 Semantic ID 对应的 Embedding 表则在下游推荐模型训练过程中进行联合优化。由于每个 Semantic ID 对应一个可训练的 Embedding，具有相近内容语义的物品可以共享部分语义表示，即使某个物品本身缺少充分的交互数据，也能够利用其内容特征生成的 Semantic ID 获得相应的语义 Embedding。这为多模态推荐模型利用内容信息改善数据稀疏物品的兴趣建模提供了一种有效途径。

总体来看，PSRQ 模型通过渐进式语义残差量化缓解了传统 RQ 在深层量化过程中原始语义信息逐渐弱化的

问题，并通过模态专属码本与模态联合码本分别保留不同模态的信息及其关联。进一步地，MCCA 网络利用目标物品的模态联合语义 Embedding 作为共享 Query，同时提取不同模态的历史兴趣，并与传统协同过滤表示进行融合。与前面多模态 SIM 模型仅将多模态 Embedding 用于历史行为检索不同，PSRQ 模型进一步将多模态内容信息转化为可学习的离散语义表示，并直接参与用户行为序列的兴趣建模。

20.6.3 MISS 模型

MISS 模型源自于快手科技发表在 CIKM 2025 会议的论文《MISS: Multi-Modal Tree Indexing and Searching with Lifelong Sequential Behavior for Retrieval Recommendation》[11]。该工作针对传统推荐系统召回阶段难以充分利用用户长期行为和物品多模态信息的问题，提出了一种结合多模态索引树与用户长期行为搜索的树形召回模型。与前面介绍的 SIM 模型主要应用于精排阶段不同，MISS 模型将长期行为序列建模直接引入树形召回框架，通过候选节点与用户历史行为之间的交互，提高召回阶段的用户兴趣建模能力。

MISS 主要包含两个核心设计：一是利用多模态 Embedding 构建索引树，使内容语义相似的物品在树结构中具有更加接近的位置；二是引入协同过滤和多模态两种行为搜索单元（General Search Unit，GSU），从用户长期行为中检索不同类型的相关兴趣，再通过精细化注意力提取用户表示。

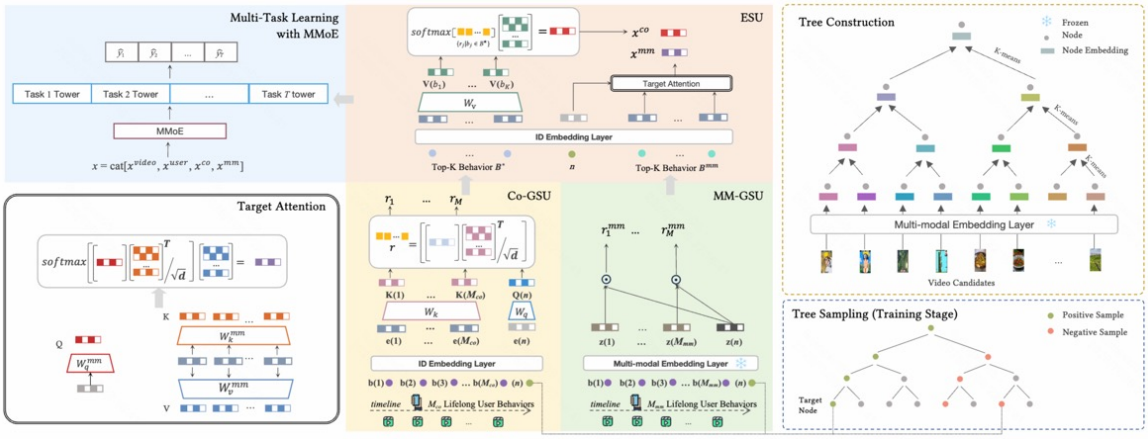


图 20.41: MISS 模型的整体模型结构。

图20.41展示了 MISS 模型的整体结构。模型首先利用多模态 Embedding 构建二叉索引树，在模型训练和召回过程中，以树节点作为当前检索目标，分别通过 Co-GSU 和 MM-GSU 检索协同过滤相关行为与多模态语义相关行为，再利用 Exact Search Unit（ESU）进行精细化兴趣建模。最终，模型将两路兴趣表示与用户、物品特征进行融合，通过多任务学习网络预测不同用户反馈目标对应的概率。

20.6.3.0.1 多模态 Embedding 与索引树构建 MISS 模型首先利用多模态 Embedding 构建树形索引结构。与传统基于物品 ID 或协同过滤 Embedding 构建索引树的方法不同，MISS 模型希望树结构能够同时反映物品之间的内容语义相似性和交互相似性。因此，MISS 模型采用基于物品对齐（Item Alignment）的多模态 Embedding 训练方法，使原本由图像和文本内容提取的 Embedding 能够进一步融入协同过滤信息。

具体来说，首先从已有的物品到物品（Item-to-Item）检索模型中选取相似物品对作为训练样本，其中相似物品作为正样本，训练批次中的其他物品作为负样本。随后，将物品的图像特征和文本特征输入多模态表示模型（Multi-modal Representation Model，MRM），获得融合内容信息的 Embedding：

$$\mathbf{c}_i = \mathrm{MRM}(\mathbf{c}_i^{text}, \mathbf{c}_i^{image}), \quad \mathbf{c}_j = \mathrm{MRM}(\mathbf{c}_j^{text}, \mathbf{c}_j^{image}) \tag{20.230}$$

为了使多模态 Embedding 能够反映物品之间的交互关系，模型采用 InfoNCE 对比学习损失进行优化：

$$\mathcal{L}_{IA} = \mathrm{InfoNCE}(\mathbf{c}_i, \mathbf{c}_j, \mathcal{J}) \tag{20.231}$$

其中，$\mathcal{J}$ 表示批次内负样本集合。通过对比学习，模型拉近相似物品对的 Embedding 距离，同时拉远负样本之

间的距离。这样得到的多模态 Embedding 不再仅仅反映图像和文本内容的相似性，还融入了 Item-to-Item 检索模型提供的协同过滤监督信息。

在获得多模态 Embedding 之后，MISS 模型利用 K-means 构建二叉索引树。在每一层中，模型将当前节点包含的物品按照多模态 Embedding 进行 K-means 二分类，并将两个聚类结果分别作为下一层的子树，持续递归划分，直到每个叶子节点只包含一个物品。这样，内容和交互表示相似的物品更有可能被组织在相同或相邻的子树中。

假设索引树的高度为 H，第 h 层的节点集合为 $\mathcal{N}_h$，每个叶子节点唯一对应一个视频物品。对于任意节点 n，记其子树中的叶子节点集合为 $\mathrm{Leaf}(n)$，则叶子节点和内部节点的多模态 Embedding 分别定义为：

$$\mathbf{z}_n = \mathbf{c}_{\pi(n)}, \quad n \in \mathcal{N}_H \tag{20.232}$$

$$\mathbf{z}_n = \mathrm{MeanPooling}\left(\{\mathbf{c}_{\pi(j)} \mid j \in \mathrm{Leaf}(n)\}\right), \quad n \notin \mathcal{N}_H \tag{20.233}$$

其中，$\pi(n)$ 表示叶子节点对应的物品 ID。对于内部节点，模型通过其子树中所有叶子物品的多模态 Embedding 进行平均池化，得到节点对应的多模态表示。这样，整棵索引树中的每个节点都具有相应的内容语义表示，可以用于后续的用户行为检索。

20.6.3.0.2 索引树学习与召回 在构建完整的多模态索引树之后，MISS 模型需要训练用户与树节点之间的匹配模型，使模型能够判断当前用户是否对某个节点子树中的物品具有兴趣。模型为每个树节点分配可训练的 ID Embedding，并构建节点预测模型 F_θ，根据用户特征、候选节点及预测任务计算对应的匹配概率。为了将物品级反馈转换为节点级监督信号，MISS 模型采用由叶子节点向上聚合的方式构造伪标签。

假设 $y^t_{\pi(i)}$ 表示用户对物品 $\pi(i)$ 在第 t 个反馈目标上的标签，则节点 n 在对应任务下的伪标签定义为：

$$\widetilde{y}^t_n = \mathbb{I}\left(\sum_{i \in \mathrm{Leaf}(n)} y^t_{\pi(i)} \geq 1\right) \tag{20.234}$$

即只要节点 n 对应的子树中存在至少一个正向反馈物品，该节点就被标记为正样本。对于每一层节点，模型保留正样本，并从没有正向反馈的节点中采样负样本，利用二分类交叉熵进行训练：

$$\mathcal{L} = \sum_{t=1}^{T} \sum_{(u,y) \in \mathcal{D}^t_{tr}} \sum_{h=1}^{H} \sum_{n \in \mathcal{S}_h(y)} \mathrm{BCE}(\widetilde{y}^t_n, \hat{y}^t_n) \tag{20.235}$$

其中，$\mathcal{S}_h(y)$ 表示第 h 层采样得到的正负节点集合，$\hat{y}^t_n$ 为模型对用户与节点在任务 t 上的预测结果：

$$\hat{y}^t_n = F_\theta(u, n; t) \tag{20.236}$$

通过这种方式，模型能够学习树节点与用户兴趣之间的匹配关系，使具有较高相关性的物品在树的上层就能够被识别出来，从而支持逐层筛选的召回过程。

在在线召回时，MISS 模型采用 Beam Search 从根节点开始逐层搜索。对于当前层已经保留的节点，模型将其子节点作为下一层的候选集合，再根据节点预测分数保留 Top-K 个节点。其过程可以表示为：

$$\mathcal{B}^{(K)}_h(u) = \mathrm{TopK}_{n \in \bigcup_{n' \in \mathcal{B}^{(K)}_{h-1}(u)} \mathcal{C}(n')} \bar{p}_F(n \mid u) \tag{20.237}$$

其中，$\mathcal{C}(n')$ 表示节点 n' 的子节点集合，$\mathcal{B}^{(K)}_h(u)$ 表示用户在第 h 层保留的 Top-K 节点集合，$\bar{p}_F$ 表示不同预测任务的综合得分。Beam Search 重复上述过程，直到搜索到叶子节点，再从最终保留的叶子节点中选择需要召回的物品：

$$\widehat{\mathcal{V}}_u = \{\pi(n) \mid n \in \mathrm{TopM}(\mathcal{B}^{(K)}_H(u))\}, \quad M \leq K \tag{20.238}$$

需要注意的是，MISS 模型并不会在模型训练完成后，利用训练得到的 ID Embedding 重新构建索引树。因为这些 ID Embedding 主要由用户交互数据训练得到，缺少原始内容信息，若使用其重新聚类，可能导致索引树丢失原有的多模态语义结构。因此，MISS 模型始终保留基于多模态 Embedding 构建的索引树，使节点组织方式与内容语义保持一致。

20.6.3.0.3 基于长期行为的多模态搜索 除了利用多模态 Embedding 构建索引树，MISS 模型还将用户长期行为序列引入树形召回过程。由于用户长期行为包含大量历史交互信息，能够反映用户较为丰富的兴趣，但同时也包含大量与当前检索目标无关的行为。如果直接对完整历史序列进行兴趣建模，不仅计算开销较高，还容易受到无关行为的干扰。

因此，MISS 模型设计了两种 General Search Unit，分别是协同过滤搜索单元（Collaborative General Search Unit，Co-GSU）和多模态搜索单元（Multi-Modal General Search Unit，MM-GSU）。二者分别利用协同过滤信息和多模态内容信息，从用户长期行为中检索与当前候选节点相关的 Top-K 个行为。随后，Exact Search Unit（ESU）对两组检索结果进行进一步的目标感知建模。

首先，对于 Co-GSU，模型将用户历史行为中的物品 ID 及当前候选节点转换为可训练的 ID Embedding。假设用户历史行为 Embedding 为 $\mathbf{E} = [\mathbf{e}_1, \ldots, \mathbf{e}_{M_{co}}]$，候选节点的 ID Embedding 为 $\mathbf{e}_n$，则通过 Target Attention 计算候选节点与各历史行为之间的相关性：

$$r_i^{co} = \frac{(\mathbf{W}_q \mathbf{e}_n)^T (\mathbf{W}_k \mathbf{e}_i)}{\sqrt{d}} \tag{20.239}$$

其中，$\mathbf{W}_q$ 和 $\mathbf{W}_k$ 分别表示 Query 和 Key 的投影矩阵，d 为 Embedding 维度。根据相关性得分，选取 Top-K 个历史行为，构成协同过滤子行为序列（Collaborative Sub-user Behavior Sequence，Co-SBS）。

与 Co-GSU 不同，MM-GSU 直接利用预训练的多模态 Embedding 进行行为检索。对于当前候选节点 n 和历史行为 b_i，分别获取其多模态 Embedding $\mathbf{z}_n$ 和 $\mathbf{z}_i$，并通过内积计算二者的相似度：

$$r_i^{mm} = \mathbf{z}_n^T \mathbf{z}_i \tag{20.240}$$

随后，根据相似度得分选取 Top-K 个历史行为，构成多模态子行为序列（Multi-Modal Sub-user Behavior Sequence，MM-SBS）。

需要注意的是，MM-GSU 中使用的多模态 Embedding 是预先训练并固定的，不参与后续检索网络的梯度更新。由于多模态 Embedding 已经通过 Item-to-Item 对齐学习了物品之间的内容及交互相似性，因此可以直接通过内积计算候选节点与历史行为之间的相关程度。这样，Co-GSU 主要捕获基于用户交互数据学习到的协同过滤关系，而 MM-GSU 则侧重于内容语义上的相似性，两种检索方式可以从不同角度提取用户长期兴趣。

在得到两组子行为序列之后，ESU 进一步对候选节点与检索结果进行精细化的 Target Attention 建模，分别得到协同过滤兴趣表示和多模态兴趣表示。对于 Co-SBS，模型可以直接复用 Co-GSU 阶段计算得到的相关性得分，并通过 Softmax 归一化得到注意力权重，再对对应的 Value 进行加权求和：

$$a_i^{co} = \mathrm{softmax}(\{r_j^{co} \mid b_j \in B^\star\})_i \tag{20.241}$$

$$\mathbf{x}_{co} = \sum_{b_i \in B^\star} a_i^{co} \mathbf{W}_v \mathbf{e}_i \tag{20.242}$$

其中，$B^\star$ 表示 Co-SBS，$\mathbf{W}_v$ 为 Value 投影矩阵。

对于 MM-SBS，模型则使用可训练的 ID Embedding 重新进行 Target Attention。这里需要区分 MM-GSU 和 ESU 两个阶段：前者使用预训练的多模态 Embedding 完成快速行为检索，而后者使用对应物品的 ID Embedding 完成精细化的兴趣建模。其注意力计算过程为：

$$a_i^{mm} = \mathrm{softmax}\left(\left\{\frac{(\mathbf{W}_q^{mm} \mathbf{e}_n)^T (\mathbf{W}_k^{mm} \mathbf{e}_j)}{\sqrt{d}} \mid \mathbf{e}_j \in \mathbf{E}'\right\}\right)_i \tag{20.243}$$

$$\mathbf{x}_{mm} = \sum_{b_i \in B^{mm}} a_i^{mm} \mathbf{W}_v^{mm} \mathbf{e}_i \tag{20.244}$$

其中，B^{mm} 表示 MM-SBS，$\mathbf{E}'$ 表示对应的历史行为 ID Embedding 序列，$\mathbf{W}_q^{mm}$、$\mathbf{W}_k^{mm}$ 和 $\mathbf{W}_v^{mm}$ 为多模态搜索分支独立使用的投影矩阵。

通过上述设计，MISS 模型利用两种不同来源的信息完成长期行为建模：Co-GSU 从协同过滤角度检索用户过去交互过的相关物品，MM-GSU 则从内容语义角度补充用户可能感兴趣的历史行为，再由 ESU 进行进一步的目标感知兴趣提取。这两种检索方式具有较好的互补性，能够缓解仅依赖协同过滤检索时难以发现内容语义相关行为的问题。

20.6.3.0.4 多任务预测与模型部署 在得到协同过滤兴趣表示 $\mathbf{x}_{co}$ 和多模态兴趣表示 $\mathbf{x}_{mm}$ 后，MISS 模型将其与用户特征、候选视频特征进行拼接，输入多门混合专家网络（Multi-gate Mixture-of-Experts，MMoE），实现多个用户反馈目标的联合建模。

假设模型包含 L_e 个 Expert 网络，对于第 t 个预测任务，利用独立的 Gate 网络计算各个 Expert 的权重，并对 Expert 输出进行加权融合：

$$\mathbf{f}^t(\mathbf{x}) = \sum_{i=1}^{L_e} g_i^t(\mathbf{x}) f_i(\mathbf{x}) \tag{20.245}$$

$$\hat{y}_n^t = h_t(\mathbf{f}^t(\mathbf{x})), \quad \mathbf{x} = \text{Concat}(\mathbf{x}_{video}, \mathbf{x}_{user}, \mathbf{x}_{co}, \mathbf{x}_{mm}) \tag{20.246}$$

其中，f_i 表示第 i 个 Expert 网络，g_i^t 表示任务 t 对应的 Gate 权重，h_t 表示任务专属的预测 Tower。模型通过前面介绍的节点级多任务交叉熵损失进行联合优化。在在线召回时，将不同任务的预测分数取平均，作为最终的节点搜索分数，用于 Beam Search 过程中的候选节点筛选。

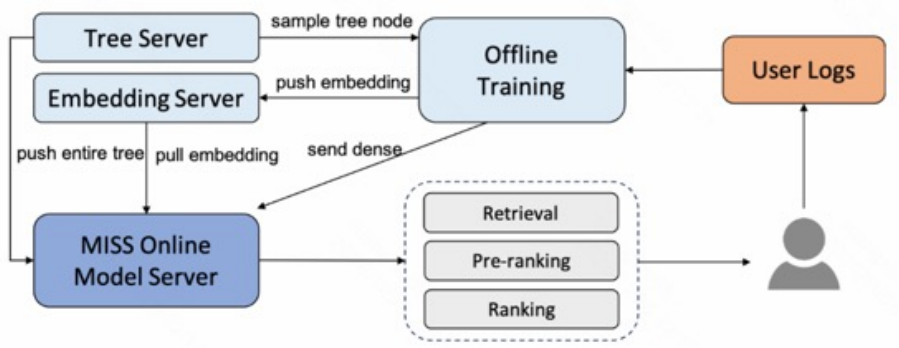


图 20.42: MISS 模型的工业部署架构。

图20.42展示了 MISS 模型在快手推荐系统中的部署架构。在离线阶段中，Tree Server 根据预训练的多模态 Embedding 构建索引树，训练系统从 Tree Server 获取树节点表示，并通过 Embedding Server 维护和更新模型所需的稀疏特征。在在线阶段中，MISS Online Model Server 在初始化时加载模型参数和完整索引树，接收到用户请求后执行 Beam Search 完成视频召回。召回结果随后与其他召回通道的候选物品一起进入粗排和精排阶段。

MISS 模型已经部署于快手推荐系统的召回阶段，并服务于超过 4 亿日活跃用户。与传统树形召回模型相比，MISS 模型通过多模态索引树改善候选物品的层次化组织方式，并利用 Co-GSU 和 MM-GSU 从不同角度搜索用户长期兴趣，使复杂的长期行为建模能够在召回阶段得到应用。

总体来看，MISS 模型的主要贡献在于将多模态内容信息和用户长期行为序列同时引入树形召回模型。一方面，多模态 Embedding 不仅用于候选物品与历史行为之间的相似度计算，还直接决定了索引树的构建方式；另一方面，Co-GSU 与 MM-GSU 分别利用协同过滤和内容语义进行行为检索，再通过 ESU 完成精细化兴趣建模。与前面介绍的多模态 SIM 模型主要改进历史行为检索表示、PSRQ 模型侧重将多模态信息转换为离散语义表示不同，MISS 模型进一步将多模态兴趣建模与树形索引结构结合，使用户长期行为能够直接参与大规模候选物品的召回过程。

20.6.4 MUSE 模型

MUSE 模型源自于阿里巴巴发表在 2025 年的论文《MUSE: A Simple Yet Effective Multimodal Search-Based Framework for Lifelong User Interest Modeling》[25]。该工作针对传统长期用户兴趣建模主要依赖 ID Embedding、难以充分利用物品内容语义的问题，系统研究了多模态信息在 General Search Unit（GSU）和 Exact Search Unit（ESU）两个阶段中的使用方式，并提出了一种简单而有效的多模态长期兴趣建模框架。

MUSE 模型的一个重要结论是：

定义 20.9

在用户行为序列建模中，不同阶段对多模态信息的使用方式存在明显差异。在 GSU 中，直接利用高质量多模态 Embedding 进行余弦相似度检索即可获得较好的效果，引入更加复杂的注意力机制或 ID 与多模态融合并不能带来明显收益。而在 ESU 中，只依赖 ID Embedding 进行 Target Attention 仍然难以充分捕获物品的内容语义，需要进一步引入多模态序列建模，并将多模态相似度与 ID Attention 进行融合。 ♣

基于这一观察，MUSE 模型采用简单的多模态行为检索方式，并在 ESU 中设计 SimTier 和 Semantic-Aware Target Attention（SA-TA）两个互补模块。图20.43展示了 MUSE 模型的整体结构。首先，模型利用 Semantic-aware Contrastive Learning（SCL）预训练物品的多模态 Embedding。在推荐模型中，GSU 通过多模态余弦相似度从用户长期历史行为中检索 Top-K 相关行为。随后，ESU 通过 SimTier 提取多模态语义兴趣分布，并通过 SA-TA 将多模态相似度融入传统 ID Attention，最终将两路兴趣表示融合后用于 CTR 预测。

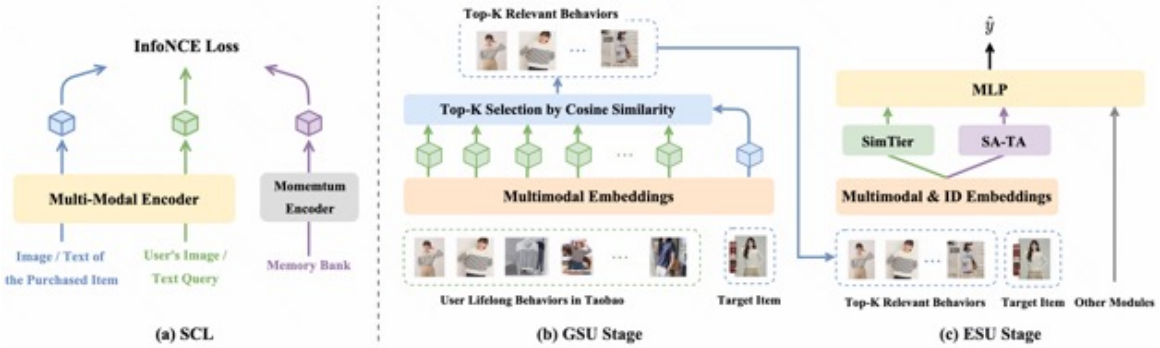


图 20.43: MUSE 模型的整体架构。

20.6.4.0.1 多模态表示与 GSU 检索 MUSE 模型首先采用预训练的多模态 Embedding 作为物品的语义表示。实验中对比了三种不同来源的 Embedding：OpenCLIP、Item-to-Item（I2I）和 Semantic-aware Contrastive Learning（SCL）。其中，OpenCLIP 通过大规模图像和文本对进行对比学习，能够提供通用的跨模态语义表示。I2I 利用用户行为日志构造物品共现正样本，并通过对比学习将协同过滤信息融入物品表示。SCL 则利用用户搜索与购买行为构造语义监督信号，例如将用户搜索时使用的图像与最终购买物品的图像构成正样本对，通过 InfoNCE 损失学习同时包含内容语义和行为相关性的多模态 Embedding。

上述三种 Embedding 均在预训练阶段完成学习，并在后续推荐模型中保持固定，通过 Embedding 查表获取，不参与 CTR 模型的梯度更新。MUSE 模型最终选择 SCL 作为主要的多模态表示，因为它不仅包含物品自身的内容信息，还通过搜索和购买行为引入了与实际推荐场景相关的语义监督。

在 GSU 阶段，MUSE 模型直接使用冻结的 SCL Embedding 计算当前目标物品与用户历史行为之间的余弦相似度。假设目标物品为 a，用户历史行为序列为 $B_u = [b_1, \ldots, b_L]$，通过 Embedding 查表分别得到目标物品表示 $\mathbf{v}_a$ 和历史行为表示序列 $\mathbf{V}_u = [\mathbf{v}_1, \ldots, \mathbf{v}_L]$，则第 i 个历史行为与目标物品之间的相似度为：

$$r_i = \frac{\mathbf{v}_a^T \mathbf{v}_i}{\|\mathbf{v}_a\|_2 \|\mathbf{v}_i\|_2} \tag{20.247}$$

根据相似度得分，选取 Top-K 个相关历史行为，构成后续 ESU 使用的子序列：

$$B_u^* = \mathrm{TopK}_{b_i \in B_u}(r_i) \tag{20.248}$$

可以看到，MUSE 模型在 GSU 中并没有引入复杂的可训练检索网络，而是直接使用预训练多模态 Embedding 的余弦相似度进行检索。通过对比实验发现，采用 ID Embedding 进行检索时，GAUC 为 0.6356。而替换为 OpenCLIP、I2I 和 SCL 后，GAUC 分别提升至 0.6365、0.6370 和 0.6377。此外，实验也进一步引入 MLP 注意力打分，或者融合 ID 相似度进行联合检索，但并没有带来明显的额外收益。上述这些结果说明，当已经拥有质量较高的多模态 Embedding 时，直接计算向量相似度就能够有效捕获目标物品与历史行为之间的语义关联，所以 GSU 并不需要承担过于复杂的建模任务。

20.6.4.0.2 SimTier：多模态语义兴趣分布建模 在完成 GSU 检索之后，MUSE 模型获得了 Top-K 相关历史行为及其对应的多模态相似度序列 $R = [r_1, \ldots, r_K]$。传统 ESU 通常直接利用历史行为的 ID Embedding 进行 Target Attention，但这种方式主要依赖物品之间的协同过滤关系，对于长尾物品和交互较少的物品，难以充分利用其内容语义。

因此，MUSE 模型引入 SimTier 模块，直接对 GSU 阶段获得的多模态相似度序列进行建模。模型将余弦相似度的取值范围 $[-1, 1]$ 均匀划分为 N 个区间（Tier），统计 Top-K 历史行为中落入各个区间的行为数量，最终形成一个 N 维直方图表示：

$$\mathbf{h}^{MM} = \text{Histogram}(R) \tag{20.249}$$

其中，$\mathbf{h}^{MM} \in \mathbb{R}^N$ 表示用户的多模态兴趣分布，每一维对应一个相似度区间内的行为数量。通过这种方式，模型能够显式刻画用户历史行为中与目标物品具有不同语义相关程度的行为分布，例如高度相关、中度相关和低度相关行为分别占据多少比例。

与直接对多模态 Embedding 序列执行 Attention 不同，SimTier 并不需要进一步处理高维的物品内容表示，而是将其与目标物品之间的相似度压缩为固定维度的统计特征。这样既保留了历史行为在语义相关性上的分布信息，也控制了后续兴趣建模的计算开销。

20.6.4.0.3 SA-TA：语义增强的目标注意力 除了通过 SimTier 显式建模多模态语义兴趣分布，MUSE 模型还进一步探索了如何将多模态信息融入传统的 ID-based Target Attention。其核心观察是：

笔记 多模态相似度和 ID Attention 具有互补性：ID Embedding 能够捕获物品之间的协同过滤关系，但对于长尾物品的泛化能力有限。而多模态 Embedding 能够提供更丰富的内容语义，但单独依赖内容相似度又难以充分反映用户实际的交互偏好。

因此，MUSE 模型提出了 Semantic-Aware Target Attention（SA-TA），将多模态相似度直接融入 ID Attention 的打分过程。假设目标物品的 ID Embedding 为 $\mathbf{e}_a \in \mathbb{R}^D$，Top-K 历史行为的 ID Embedding 矩阵为 $\mathbf{E}_u \in \mathbb{R}^{K \times D}$，传统 Target Attention 首先计算目标物品与各历史行为之间的相关性：

$$\boldsymbol{\alpha}^{ID} = \frac{(\mathbf{E}_u \mathbf{W}^k)(\mathbf{e}_a \mathbf{W}^q)^T}{\sqrt{D}} \tag{20.250}$$

与此同时，GSU 阶段已经计算得到了目标物品与 Top-K 历史行为之间的多模态相似度向量 $\mathbf{R} = [r_1, \ldots, r_K]$。SA-TA 将两种信号进行融合，构造新的注意力打分：

$$\boldsymbol{\alpha}^{Fusion} = \gamma_1 \boldsymbol{\alpha}^{ID} + \gamma_2 \mathbf{R} + \gamma_3 (\boldsymbol{\alpha}^{ID} \odot \mathbf{R}) \tag{20.251}$$

其中，γ_1、γ_2 和 γ_3 为可学习的标量参数，$\odot$ 表示逐元素相乘。融合后的注意力权重不仅保留了 ID Embedding 所反映的协同过滤关系，还直接引入了目标物品与历史行为之间的多模态语义相似度。其中，交叉项 $\boldsymbol{\alpha}^{ID} \odot \mathbf{R}$ 进一步刻画了两种相关性信号之间的交互。最后，模型通过 Softmax 对融合后的打分进行归一化，并对历史行为的 Value 进行加权求和，得到语义增强的长期兴趣表示：

$$\mathbf{u}_l^{ID} = \text{Softmax}(\boldsymbol{\alpha}^{Fusion})^T (\mathbf{E}_u \mathbf{W}^v) \tag{20.252}$$

图20.44展示了 SA-TA 模块的主要计算过程。模型一方面通过 ID Embedding 计算传统 Target Attention，另一方面利用多模态 Embedding 获得目标物品与历史行为之间的语义相似度，最后将两种打分进行融合，得到语义增强的注意力权重。最终，MUSE 模型将 SimTier 得到的多模态兴趣表示与 SA-TA 得到的 ID 兴趣表示进行拼接，形成完整的长期用户兴趣表示：

$$\mathbf{u}_l = \mathbf{h}^{MM} \oplus \mathbf{u}_l^{ID} \tag{20.253}$$

该表示与其他用户及目标物品特征一起输入后续 CTR 预测网络，完成最终的点击率预测。

通过上述设计，SimTier 和 SA-TA 分别从不同角度利用多模态信息。SimTier 直接统计目标物品与历史行为之间的语义相关性分布，而 SA-TA 则将这种语义相关性融入协同过滤注意力计算。对比实验也验证了两种模块的互补性。在相同 GSU 和 SCL Embedding 条件下，传统 Target Attention 的 GAUC 为 0.6301，SimTier 单独使用时达到 0.6345，而将 SimTier 与 SA-TA 结合后达到 0.6377，相比原始 Target Attention 提升了 1.21%。这说明仅

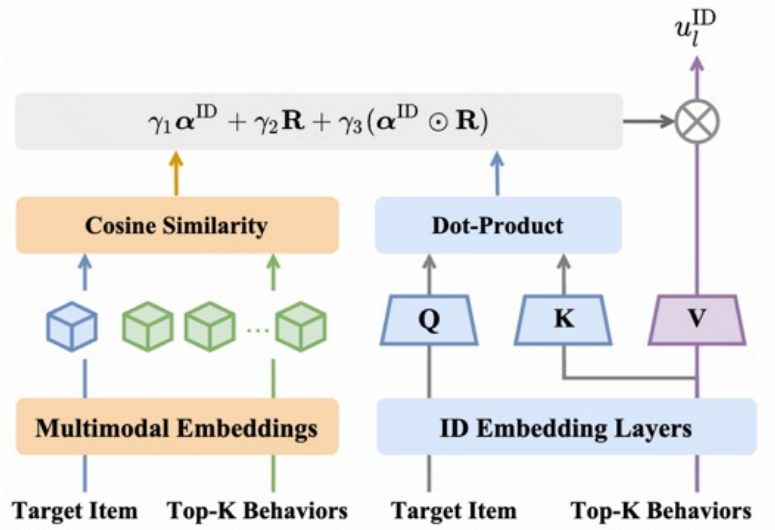


图 20.44: SA-TA 的计算结构。

仅在 GSU 中利用多模态 Embedding 进行行为检索还不够，在 ESU 中进一步建模多模态语义及其与 ID 信号的关联，也能够带来额外收益。

20.6.4.0.4 模型部署与工程优化 MUSE 模型已经部署于淘宝广告系统的精排阶段，能够对最长约 100K 的用户历史行为进行建模。由于用户长期行为序列及其对应的多模态 Embedding 需要从远程存储中获取，如果将这一过程放在精排请求的关键路径上，网络通信和特征读取可能成为主要的延迟瓶颈。

因此，MUSE 模型采用异步预取（Pre-fetching）策略，将 GSU 的数据获取过程与推荐系统的召回（Matching）阶段并行执行。具体来说，GSU 服务器提前从用户行为存储和多模态 Embedding 服务中获取用户历史行为及对应 Embedding，并将这些数据缓存到 GPU 显存中。由于这一过程与召回阶段并行，通常可以在候选物品进入精排之前完成，从而将数据读取的延迟隐藏在原有推荐链路中。

当候选物品进入精排阶段之后，模型直接使用已经缓存的用户行为 Embedding 计算多模态相似度并完成 Top-K 检索，再将检索结果交给 ESU 中的 SimTier 和 SA-TA 进行精细化兴趣建模。与此同时，GSU 的相似度计算和 Top-K 选择还可以与精排中的其他特征处理过程并行执行，进一步减少额外的在线延迟。

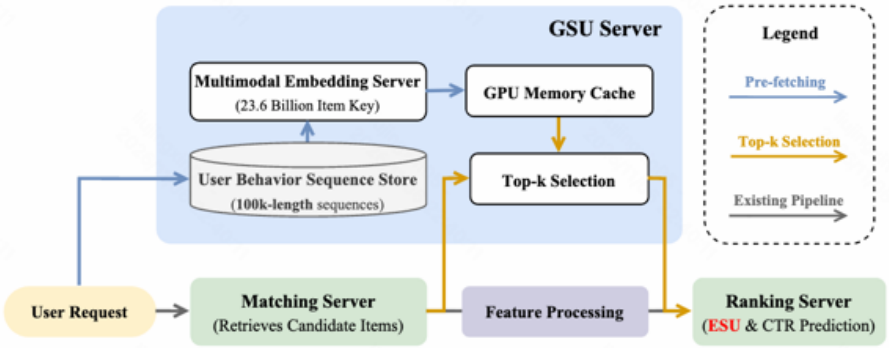


图 20.45: MUSE 模型在淘宝展示广告系统中的部署架构。

图20.45展示了 MUSE 模型的线上部署架构。通过异步预取和 GPU 缓存，模型将用户长期行为及多模态 Embedding 的读取过程前置，使得精排阶段能够直接使用缓存数据完成行为检索和后续兴趣建模。通过这种工程设计，MUSE 模型将超长行为序列的数据获取从精排关键路径中分离出来，使 GSU 能够利用更长的用户历史行为，同时控制在线计算与通信成本。

总体来看，MUSE 模型在多模态长期兴趣建模中的主要特点，是针对 GSU 和 ESU 采用了不同的多模态信息利用方式。GSU 通过高质量多模态 Embedding 的简单相似度检索获得相关历史行为，ESU 则利用 SimTier 显

式建模语义相关性分布，并通过 SA-TA 将多模态语义与协同过滤信号融合。与前面介绍的多模态 SIM 模型主要改进检索表示、PSRQ 模型侧重离散语义表示学习、MISS 模型将多模态信息引入树形召回不同，MUSE 模型进一步将多模态建模同时扩展到长期行为检索和精细化兴趣提取两个阶段，并通过异步预取解决超长序列在精排阶段的工程开销问题。

20.7 基于时序图的用户行为序列建模

前面介绍的用户行为序列建模方法主要通过序列编码、注意力计算或者历史行为检索提取用户兴趣。然而，用户的历史交互不仅具有时间顺序，还隐含着复杂的关联关系。例如，用户可能在不同时间与多个相似物品发生交互，也可能在不同场景下表现出截然不同的兴趣偏好。仅按照时间顺序对历史行为进行建模，难以充分利用这些高阶关联信息。

图神经网络为建模用户行为之间的复杂关系提供了另一种思路。通过将用户、物品或行为上下文组织为图结构，模型可以利用节点之间的连接关系进行信息传播，从而捕获不同历史行为之间的高阶关联。在此基础上，结合交互发生的时间信息，还可以进一步描述用户兴趣随时间的演化过程。

本节介绍两种具有代表性的方法。DisenCTR 模型通过构建动态用户—物品交互图，将用户兴趣解耦为多个潜在兴趣分量，并利用 Mixture of Hawkes Process 建模不同兴趣下历史行为的时间激励效应。CoFARS 模型则从用户行为发生的上下文出发，利用上下文偏好分布识别相似行为场景，并通过时序图建模不同上下文之间的关联，从而实现面向目标上下文的长期行为序列筛选。

20.7.1 DisenCTR 模型

DisenCTR 模型源自于美团发表在 SIGIR 2022 会议的论文《DisenCTR: Dynamic Graph-based Disentangled Representation for Click-Through Rate Prediction》[24]。该工作针对传统序列建模方法难以同时捕获用户多样化兴趣及其动态演化的问题，提出了一种基于动态交互图的多兴趣 CTR 预测模型。其核心思想是利用用户与物品之间随时间演化的高阶交互关系，将用户和物品的 Embedding 解耦为多个兴趣分量，再通过图上的信息传播不断细化各个兴趣表示，最终结合 Hawkes 过程建模历史行为对当前交互的时间影响。

20.7.1.0.1 动态时序交互图构建 传统序列建模方法通常直接按照时间顺序组织用户历史行为，将用户与物品之间的交互表示为一条行为序列。然而，仅利用单个用户的历史交互，难以充分获取其他用户与相同物品之间的关联信息。实际上，不同用户可能通过共同交互的物品形成间接关联，而这些高阶关系能够为用户兴趣建模提供额外的协同过滤信号。

因此，DisenCTR 模型构建了一个动态演化的用户—物品二部图，并利用广度优先搜索（Breadth-First Search，BFS）从目标用户出发，逐层扩展其历史交互邻居。假设用户集合和物品集合分别为 $\mathcal{U}$ 和 $\mathcal{V}$，所有交互记录构成集合 $\mathcal{E} = \{(u, v, \tau)\}$，其中 τ 表示交互发生的时间。对于目标用户 u 和当前预测时间 t，模型将其初始化为第 0 跳节点，并将用户的历史交互作为第 1 跳邻居。随后，根据用户—物品交互关系交替扩展用户节点和物品节点，构建更高阶的历史交互关系。

具体来说，第 $l+1$ 跳的交互集合定义为：

$$\mathcal{G}_{u,t}^{(l+1)} = \{(u', v, \tau) \mid \tau < t',\ (u', v, \tau) \in \mathcal{E},\ (\cdot, u', t') \in \mathcal{G}_{u,t}^{(l)}\} \tag{20.254}$$

其中，$\tau < t'$ 表示新扩展的邻居交互必须早于当前中心节点对应的交互时间，以保证图中的信息传播符合历史时间顺序。通过逐层扩展，可以获得目标用户在当前时间之前的多跳交互关系，最终形成动态时序图：

$$\mathcal{G}_{u,t} = \{\mathcal{G}_{u,t}^{(0)}, \mathcal{G}_{u,t}^{(1)}, \ldots, \mathcal{G}_{u,t}^{(L)}\} \tag{20.255}$$

其中，L 表示图的扩展层数。与普通用户—物品二部图不同，这种构建方式不仅保留了用户和物品之间的高阶连接关系，还通过时间约束保证不同层级的历史交互具有明确的先后顺序，从而能够同时利用图结构和序列中的时间信息。

20.7.1.0.2 动态图解耦表示学习 在得到动态时序交互图之后，DisenCTR 模型进一步利用图神经网络提取用户的多样化兴趣。其核心模块为动态图解耦层（Dynamic Graph Disentangling Layer），主要包含解耦 Embedding 变换、解耦 Embedding 传播以及兴趣意图迭代更新三个部分。

首先，模型将用户和物品的原始特征映射到多个独立的潜在空间，将原本单一的 Embedding 表示拆分为 K 个兴趣分量。以用户 u 为例，其第 k 个兴趣分量表示为：

$$\mathbf{c}_{u,k} = \sigma(\mathbf{W}_k \mathbf{x}_u) \tag{20.256}$$

其中，$\mathbf{x}_u$ 表示用户原始特征，$\mathbf{W}_k$ 表示第 k 个兴趣分量对应的映射矩阵，$\sigma(\cdot)$ 表示激活函数。类似地，物品 v 也通过独立的映射矩阵得到多个兴趣分量 $\mathbf{c}_{v,k}$。经过上述变换，用户和物品的表示不再局限于单个 Embedding，而是被分解为多个潜在兴趣空间，从而为后续不同兴趣下的图信息传播提供基础。

接下来，模型需要确定用户与物品之间的交互应该由哪个兴趣分量解释。因此，DisenCTR 模型针对每个兴趣分量维护一个意图评分矩阵 S_k，其中 $S_k(u,v)$ 表示用户 u 与物品 v 在第 k 个兴趣下的关联程度。在初始时，各个兴趣分量的评分相同，随后通过图传播进行迭代更新。对于第 i 次迭代，首先在不同兴趣分量之间进行 Softmax 归一化，得到交互对应的兴趣分布：

$$\widetilde{S}_k^i(u,v) = \frac{\exp(S_k^i(u,v))}{\sum_{k'=1}^{K} \exp(S_{k'}^i(u,v))} \tag{20.257}$$

该分布可以看作用户与物品之间的兴趣分配权重，反映了当前交互主要由哪些兴趣驱动。利用这些权重，模型分别在不同兴趣分量下进行邻居信息传播。以用户节点为例，其第 k 个兴趣分量的更新过程为：

$$\mathbf{c}_{u,k}^i = \sum_{v \in \mathcal{N}_u} \frac{\widetilde{S}_k^i(u,v)}{\sqrt{D_k^t(u) D_k^t(v)}} \mathbf{c}_{v,k}^{i-1} \tag{20.258}$$

其中，$\mathcal{N}_u$ 表示用户 u 的物品邻居集合，$D_k^t(u)$ 和 $D_k^t(v)$ 分别为当前时间下用户节点和物品节点在第 k 个兴趣分量上的归一化度数：

$$D_k^t(u) = \sum_{v' \in \mathcal{N}_u} \widetilde{S}_k^i(u,v'), \quad D_k^t(v) = \sum_{u' \in \mathcal{N}_v} \widetilde{S}_k^i(u',v) \tag{20.259}$$

这种传播方式能够在每个兴趣空间内聚合相关邻居的信息，同时利用归一化权重控制不同节点邻居数量带来的影响。物品节点则采用对称的方式聚合用户邻居信息。通过多轮传播，用户和物品的各个兴趣分量能够逐渐融合高阶交互关系。

此外，DisenCTR 模型并不是在初始化时就固定用户与物品之间的兴趣分配，而是根据每次图传播后的表示持续更新意图评分矩阵：

$$S_k^{i+1}(u,v) = S_k^i(u,v) + (\mathbf{c}_{u,k}^i)^T \tanh(\mathbf{c}_{v,k}^i) \tag{20.260}$$

其中，更新项用于衡量用户和物品在第 k 个兴趣空间中的关联程度。经过多轮迭代之后，模型能够逐渐增强与当前兴趣相关的用户—物品连接，并利用这些关联关系进一步细化各个兴趣分量。

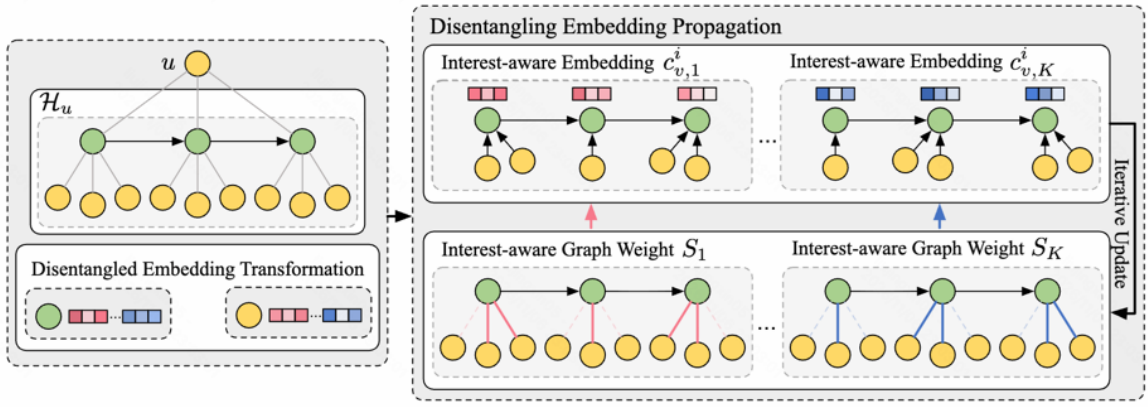


图 20.46: DisenCTR 的动态图解耦过程。

图20.46展示了动态图解耦层的整体过程。模型首先将用户和物品表示拆分为多个兴趣分量，然后分别在不同兴趣空间中进行图传播，并根据更新后的节点表示迭代调整兴趣关联权重。通过堆叠多个动态图解耦层，DisenCTR 模型能够将多跳交互关系逐步融入不同兴趣分量，使用户和物品的表示同时包含多样化兴趣及其高阶协同关系。

20.7.1.0.3 基于 Mixture of Hawkes Process 的 CTR 预测 经过动态图解耦之后，模型已经获得用户和物品在不同兴趣空间中的表示。然而，用户兴趣不仅具有多样性，还会随着历史交互不断变化。例如，用户近期对某类物品的连续点击可能会进一步增强其短期兴趣，而随着时间推移，这些行为对当前决策的影响通常会逐渐减弱。

因此，DisenCTR 模型进一步引入 Hawkes 过程，建模不同兴趣下历史交互对当前预测的时间激励效应。Hawkes 过程是一种典型的时间点过程，通过条件强度函数描述历史事件对未来事件发生概率的影响。DisenCTR 模型将用户与物品的交互视为由多个潜在兴趣共同驱动的过程，针对每个兴趣分量分别构建条件强度函数，再将多个兴趣分量的结果进行融合。

图20.47展示了基于 Mixture of Hawkes Process（MHP）的 CTR 预测结构。模型首先利用动态图解耦层得到当前用户和目标物品的多兴趣表示，再分别计算各个兴趣对应的基础强度和历史行为激励项，最终融合不同兴趣下的条件强度得到 CTR 预测结果。

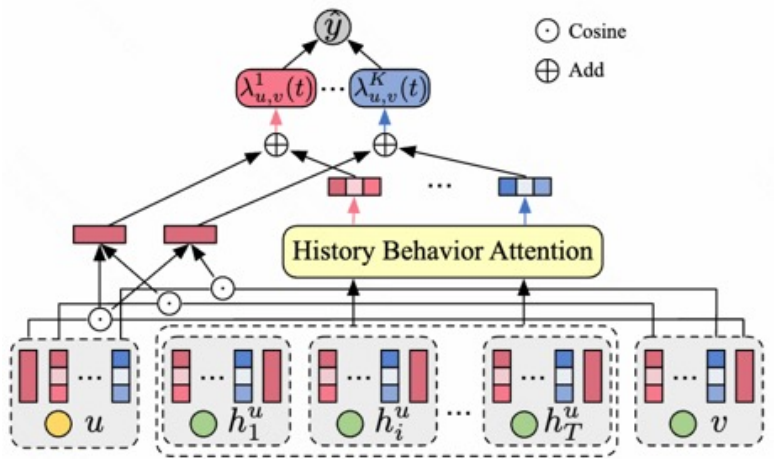


图 20.47: DisenCTR 模型的多兴趣 Hawkes 预测结构。

对于目标用户 u 和目标物品 v，模型首先利用原始 ID Embedding 计算基础强度 $\mu_{u,v}$，并结合最终图传播得到的兴趣表示，计算第 k 个兴趣分量对应的强度系数：

$$\mu_{u,v} = \mathcal{F}(\widetilde{\mathbf{e}}_u, \widetilde{\mathbf{e}}_v), \quad \gamma^k_{u,v} = \text{Softplus}(\mathcal{F}(\mathbf{c}^I_{u,k}, \mathbf{c}^I_{v,k})) \tag{20.261}$$

其中，$\mathcal{F}(\cdot)$ 表示余弦相似度，$\widetilde{\mathbf{e}}_u$ 和 $\widetilde{\mathbf{e}}_v$ 分别表示用户和物品的原始 ID Embedding，I 表示最终的图传播迭代次数。Softplus 用于保证兴趣强度系数非负。

在此基础上，模型将用户历史行为对当前交互的影响纳入条件强度函数。对于用户历史行为 $h \in \mathcal{H}_u$，首先根据历史物品与目标物品之间的 ID 相似度，以及二者在兴趣分量 k 上的关联程度，计算历史行为的归一化权重：

$$\alpha^k_{v,h} = \frac{\exp(\mathcal{F}(\widetilde{\mathbf{e}}_v, \widetilde{\mathbf{e}}_h)\gamma^k_{v,h})}{\sum_{h' \in \mathcal{H}_u} \exp(\mathcal{F}(\widetilde{\mathbf{e}}_v, \widetilde{\mathbf{e}}_{h'})\gamma^k_{v,h'})} \tag{20.262}$$

$$\gamma^k_{v,h} = \text{Softplus}(\mathcal{F}(\mathbf{c}^I_{v,k}, \mathbf{c}^I_{h,k})) \tag{20.263}$$

随后，模型引入时间衰减核函数：

$$J(t-\tau) = \exp(-D_u(t-\tau)) \tag{20.264}$$

其中，$D_u(\cdot)$ 是用户相关的可训练时间衰减函数，τ 表示历史行为发生的时间。随着当前时刻与历史交互时间间

隔增大，历史行为对当前交互的激励作用会逐渐减弱。最终，第 k 个兴趣分量下的条件强度函数可以表示为：

$$\lambda_{u,v}^{k}(t) = \mu_{u,v}\gamma_{u,v}^{k} + \sum_{h\in\mathcal{H}_u} \alpha_{v,h}^{k}\gamma_{v,h}^{k} J(t-\tau_h) \tag{20.265}$$

其中，第一项表示当前用户与目标物品在对应兴趣下的基础交互强度，第二项表示历史行为经过兴趣关联加权和时间衰减后，对当前交互产生的激励作用。

最后，模型将不同兴趣分量下的条件强度取平均，并通过 Sigmoid 函数得到 CTR 预测结果：

$$\hat{y}_{u,v} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\lambda_{u,v}^{k}(t)\right) \tag{20.266}$$

模型采用二元交叉熵损失进行优化：

$$\mathcal{L} = -\sum_{(u,v,t)\in\mathcal{D}} [y\log\hat{y}_{u,v} + (1-y)\log(1-\hat{y}_{u,v})] \tag{20.267}$$

通过上述设计，DisenCTR 模型将多兴趣图表示学习与时间点过程结合起来：动态图解耦层负责从高阶交互关系中提取用户的多样化兴趣，MHP 则进一步描述不同兴趣下历史行为的时间激励和衰减效应，从而同时建模用户兴趣的多样性与动态性。

20.7.2 CoFARS 模型

CoFARS 模型源自于美团发表在 WWW 2024 会议的论文《Context-based Fast Recommendation Strategy for Long User Behavior Sequence in Meituan Waimai》[10]。与 DisenCTR 模型直接利用用户—物品交互图进行多兴趣表示学习不同，CoFARS 模型关注用户行为发生时的上下文信息，认为用户在不同场景下可能表现出明显不同的兴趣偏好。例如，在外卖推荐场景中，同一个用户在工作日午餐、周末晚餐、公司和家庭等不同上下文中，可能具有截然不同的品类和价格偏好。

基于这一观察，CoFARS 模型提出了一种基于上下文的长期行为子序列筛选方法。它不再针对每个候选物品单独检索历史行为，而是首先识别与当前目标上下文具有相似用户偏好的历史上下文，再从这些上下文中筛选对应的历史交互物品。为了解决上下文偏好表示和时间关系建模问题，CoFARS 模型引入了概率编码器（Probability Encoder）和基于图的时序聚合模块（Graph-based Temporal Aggregator）。

图20.48展示了 CoFARS 模型的整体流程。模型首先根据用户历史交互构建上下文序列，并通过概率编码器学习上下文偏好表示及偏好原型。随后，模型会构建包含上下文节点和原型节点的时序图，通过图注意力网络聚合历史上下文之间的时间关系；最后，模型会结合用户近期行为和目标上下文识别相关原型，并根据原型筛选历史行为子序列，供后续 Target Attention 进行精细化兴趣建模。

20.7.2.0.1 基于偏好分布的上下文建模 在美团外卖推荐场景中，用户的交互行为通常受到地理位置、用餐时间、天气和节假日等上下文因素的影响。传统方法可以通过上下文 Embedding 之间的余弦相似度识别相似场景，但这种方式主要依赖 Embedding 的表示质量，难以直接保证两个上下文在用户偏好层面具有一致性。

CoFARS 模型认为，相比起直接比较上下文的隐向量，通过用户在不同上下文中对物品属性的偏好分布来衡量场景相似性更加直观。例如，在公司午餐场景下，用户可能更偏好价格适中、出餐较快的快餐类物品。而在家庭晚餐场景下，则可能更倾向于选择价格较高、适合多人用餐的餐饮类别。如果两个上下文中的物品属性分布相近，就可以认为用户在这两个场景下具有相似的偏好。

假设 $\mathcal{A}$ 表示物品属性及其离散取值的组合空间，包括品类、价格等属性。对于上下文 C_i，利用历史交互日志统计其对应的物品属性分布，记为 $D(C_i)$。为了衡量两个上下文之间的偏好差异，CoFARS 模型采用对称化的 KL 散度进行度量：

$$\mathrm{KL}(C_i, C_j) = \sum_{\mathbf{a}\in\mathcal{A}} D(C_i)(\mathbf{a})\log\frac{D(C_i)(\mathbf{a})}{D(C_j)(\mathbf{a})} \tag{20.268}$$

$$\mathrm{JS}(C_i, C_j) = \frac{1}{2}\left(\mathrm{KL}(C_i, C_j) + \mathrm{KL}(C_j, C_i)\right) \tag{20.269}$$

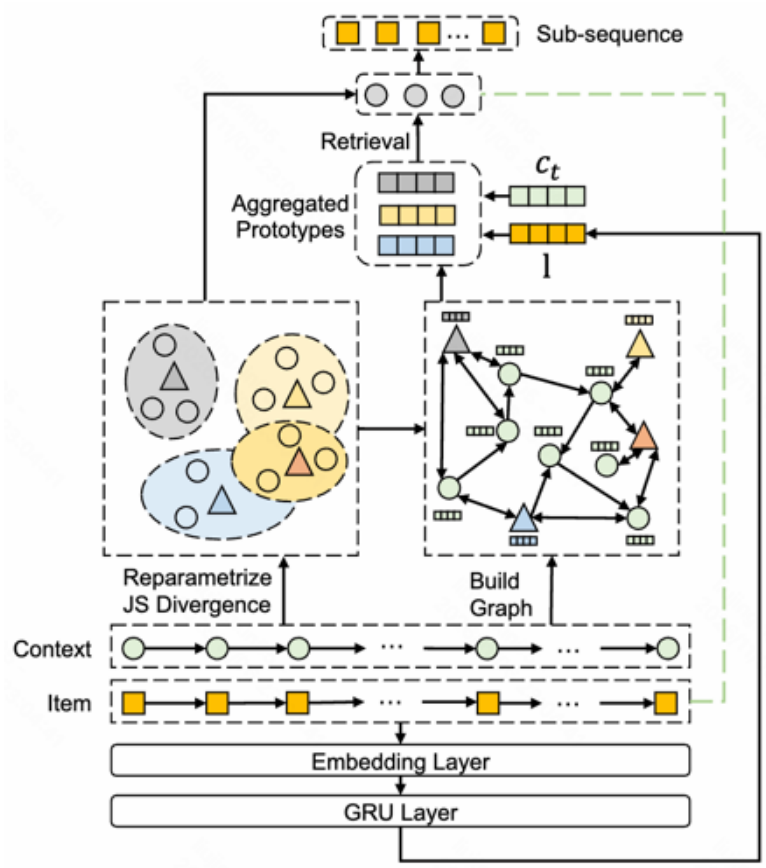


图 20.48: CoFARS 模型的整体框架。

这里沿用论文对该指标的命名。需要注意的是，按照上述定义，它实际上是对称 KL 散度，而不是通常以中间分布为基础定义的标准 Jensen-Shannon 散度。该指标越小，表示两个上下文下的物品属性分布越接近，用户偏好也越相似。

然而，如果直接针对每个目标上下文遍历全部历史上下文并计算偏好分布差异，不仅需要大量计算，还面临上下文冷启动问题。此外，不同目标上下文能够匹配到的历史上下文数量也存在较大差异，容易导致筛选后的行为序列过长或过短。因此，CoFARS 模型进一步引入偏好原型（Preference Prototypes），将具有相似偏好的上下文映射到有限数量的原型中。

假设原型集合为 $\mathcal{O} = \{o_1, o_2, \ldots, o_{|\mathcal{O}|}\}$，每个原型代表偏好空间中的一个中心，用于组织具有相似用户偏好的上下文。这样，模型不需要直接对当前目标上下文与所有历史上下文进行匹配，而是可以先找到与其偏好相似的原型，再通过原型关联的上下文筛选历史行为。

由于原型本身只有可训练的隐向量，无法直接从日志中获得对应的物品属性分布。因此，CoFARS 模型设计了概率编码器，将上下文和原型的隐向量映射为物品属性的概率表示。对于上下文 C_i，利用 $\widehat{\mathbf{c}}_i$ 表示其全局偏好 Embedding，再与用户 Embedding$\mathbf{u}$ 相加得到个性化上下文表示：

$$\mathbf{c}_i = \mathbf{u} + \widehat{\mathbf{c}}_i, \quad P(\mathbf{c}_i) = \mathrm{MLP}(\mathbf{c}_i) \tag{20.270}$$

其中，MLP 的输出维度等于所有物品属性离散取值数量之和，并通过 Sigmoid 函数映射为属性偏好概率。对于任意两个隐向量，模型首先根据其概率编码计算对应的 KL 散度，再得到估计的对称偏好差异：

$$\widetilde{\mathrm{JS}}(\mathbf{c}_i, \mathbf{c}_j) = \frac{1}{2}\left(\mathrm{KL}_f(P(\mathbf{c}_i), P(\mathbf{c}_j)) + \mathrm{KL}_f(P(\mathbf{c}_j), P(\mathbf{c}_i))\right) \tag{20.271}$$

其中，KL_f 表示在概率编码结果上计算的 KL 散度。

为了使隐空间中的偏好差异与真实交互日志中的属性分布差异保持一致，CoFARS 模型通过均方误差约束

两者：

$$\mathcal{L}_{MSE} = \frac{1}{|\mathcal{C}|^2} \sum_{i=1}^{|\mathcal{C}|} \sum_{j=1}^{|\mathcal{C}|} \left(\mathrm{JS}(C_i, C_j) - \widetilde{\mathrm{JS}}(\mathbf{c}_i, \mathbf{c}_j) \right)^2 \tag{20.272}$$

通过最小化该损失，模型能够学习将上下文隐向量映射为具有实际偏好含义的属性概率分布，并据此计算不同上下文之间的相似性：

$$\mathrm{sim}(\mathbf{c}_i, \mathbf{c}_j) = 1 - \widetilde{\mathrm{JS}}(\mathbf{c}_i, \mathbf{c}_j) \tag{20.273}$$

对于偏好原型，模型采用类似方式构造其个性化表示 $\mathbf{o}_i = \mathbf{u} + \widehat{\mathbf{o}}_i$，并进一步引入原型独立性损失，避免所有原型集中到相似的偏好区域：

$$\mathcal{L}_{IND} = -\frac{1}{|\mathcal{O}|^2} \sum_{i=1}^{|\mathcal{O}|} \sum_{j=1}^{|\mathcal{O}|} \widetilde{\mathrm{JS}}(\mathbf{o}_i, \mathbf{o}_j) \tag{20.274}$$

由于该损失采用负的原型间偏好差异，最小化它能够促使不同原型之间保持较大的偏好距离，从而提高原型集合对多样化用户偏好的覆盖能力。

20.7.2.0.2 基于图的时序聚合 概率编码器能够识别偏好相似的上下文，但如果只利用上下文之间的偏好分布进行匹配，仍然无法充分保留用户历史行为中的时间演化信息。例如，用户在不同用餐时间可能形成相似的物品偏好，但这些上下文之间的发生顺序也可能反映用户行为的周期性和场景转换规律。

因此，CoFARS 模型进一步构建基于上下文的时序图。与直接在物品之间建立共现图不同，CoFARS 模型将用户历史交互中的上下文作为图节点，将历史行为序列中相邻上下文之间的转移关系作为边。由于上下文类型相对于物品更加有限，而且用户的长期行为中可能反复出现相同的时间、地点等场景组合，因此上下文图能够形成较为丰富的重复连接关系，更适合利用图神经网络进行信息聚合。

在此基础上，CoFARS 模型还将偏好原型加入图中，使得原型节点能够与偏好相近的上下文节点建立连接。为了避免每个原型与所有上下文完全连接所带来的图规模膨胀，模型根据原型与上下文之间的相似度进行边筛选。模型计算出二者的偏好相似度如下所示：

$$\beta_{ij} = \mathrm{sim}(\mathbf{o}_i, \mathbf{c}_j) \tag{20.275}$$

随后，模型采用 Gumbel-Softmax 对原型和上下文之间的离散连接关系进行可微分建模，使得模型能够在训练过程中学习是否建立对应的图边。在经过筛选后，相似度较高的原型与上下文节点之间建立双向连接。

最终，图中的节点包括所有偏好原型和历史上下文，其初始表示为：

$$\mathbf{H}^0 = [\mathbf{o}_1; \ldots; \mathbf{o}_{|\mathcal{O}|}; \mathbf{c}_1; \ldots; \mathbf{c}_{|\mathcal{C}|}] \tag{20.276}$$

在此基础上，利用图注意力网络（Graph Attention Network，GAT）对节点邻居进行信息聚合。对于第 l 层中的节点 i 和其邻居 j，首先计算二者之间的注意力权重：

$$\alpha_{ij}^l = \frac{\exp(\mathrm{LeakyReLU}(\mathrm{sim}(\mathbf{H}_i^l \mathbf{W}^l, \mathbf{H}_j^l \mathbf{W}^l)))}{\sum_{k \in \mathcal{N}_i} \exp(\mathrm{LeakyReLU}(\mathrm{sim}(\mathbf{H}_i^l \mathbf{W}^l, \mathbf{H}_k^l \mathbf{W}^l)))} \tag{20.277}$$

随后，根据注意力权重聚合邻居节点表示：

$$\mathbf{H}_i^{l+1} = \mathrm{ReLU}\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^l \mathbf{H}_j^l \mathbf{W}^l \right) \tag{20.278}$$

其中，$\mathcal{N}_i$ 表示节点 i 的前驱邻居集合，$\mathbf{W}^l$ 表示第 l 层的变换矩阵。经过多层图传播之后，可以得到融入上下文转移信息的原型表示 $\widetilde{\mathbf{o}}_i$ 和上下文表示 $\widetilde{\mathbf{c}}_j$。这样，原本仅由上下文偏好决定的节点表示进一步包含了历史行为中的时序关联，为后续的目标上下文匹配提供更加丰富的信息。

20.7.2.0.3 目标上下文匹配与行为序列筛选 在获得融合时序信息的原型和上下文表示之后，CoFARS 模型需要根据当前请求的目标上下文，确定用户当前可能具有的兴趣偏好，并从历史行为中筛选相关子序列。首先，模

型利用用户最近的 r 次交互行为构建短期行为序列，并通过 GRU 提取用户近期兴趣表示：

$$\mathbf{l} = \mathrm{GRU}(\mathbf{e}_{n-r}, \mathbf{e}_{n-r+1}, \ldots, \mathbf{e}_n) \tag{20.279}$$

其中，$\mathbf{e}_i$ 表示第 i 次历史交互对应的上下文 Embedding，r 远小于完整历史序列长度 n。通过近期行为，模型能够获得用户当前阶段的短期偏好，弥补仅依赖长期上下文统计信息所带来的不足。

假设当前请求对应的目标上下文为 C_t，模型利用目标上下文表示、时序聚合后的原型表示以及短期兴趣，计算每个偏好原型与当前请求之间的匹配程度：

$$z_i^t = \sigma\left(\mathrm{sim}(\widetilde{\mathbf{o}}_i, \widetilde{\mathbf{c}}_t) \cdot (\widetilde{\mathbf{o}}_i^T \mathbf{l})\right) \tag{20.280}$$

其中，z_i^t 表示目标上下文与第 i 个偏好原型的匹配概率。该计算同时考虑了原型与目标上下文之间的偏好相似性，以及原型与用户近期兴趣之间的关联程度。通过这种方式，模型可以避免仅根据目标上下文的静态特征进行匹配，而是结合用户近期行为判断当前更可能激活哪些长期偏好原型。

在得到相关原型之后，CoFARS 模型根据原型所关联的历史上下文，进一步筛选用户在这些上下文下发生交互的物品，形成目标场景相关的行为子序列。后续 CTR 和 CTCVR 预测模型再对该子序列执行 Target Attention，以提取与具体候选物品相关的兴趣表示。需要注意的是，CoFARS 模型的子序列筛选过程不依赖具体候选物品，而是根据用户当前的目标上下文和偏好原型完成。这与 SIM、ETA 等针对每个候选物品分别检索相关历史行为的方法不同，可以避免在同一个请求中对完整历史行为序列重复执行候选物品相关的检索。

在训练过程中，模型通过候选物品预测损失以及前面介绍的两个辅助约束共同优化。假设 $\widetilde{\mathcal{O}}$ 表示用于预测的原型集合，v 表示目标 PoI，模型根据原型与目标上下文的匹配结果以及原型表示计算预测分数：

$$s_{i,v}^t = z_i^t \cdot (\widetilde{\mathbf{o}}_i^T \mathbf{v}) \tag{20.281}$$

其中，$\mathbf{v}$ 表示目标 PoI 的 Embedding。模型通过二元交叉熵构造推荐损失，并结合上下文偏好分布对齐损失和原型独立性损失进行联合优化：

$$\mathcal{L} = \mathcal{L}_{REC} + \gamma \mathcal{L}_{MSE} + \lambda \mathcal{L}_{IND} \tag{20.282}$$

其中，γ 和 λ 分别用于控制两个辅助损失的权重。通过这种联合优化，模型既能够学习当前上下文与偏好原型之间的匹配关系，也能够保证原型表示能够反映真实的物品属性偏好，并覆盖不同类型的用户兴趣。

对于在线推理环节，已经出现过的上下文与偏好原型之间的相似度可以提前计算和存储。当存在冷启动上下文时，模型可以通过概率编码器生成新的上下文表示，再计算其与各个偏好原型之间的相似度。最终，根据匹配的原型找到关联的历史上下文，并提取对应的 PoI 行为子序列，再交由后续 Target Attention 模块完成精细化兴趣建模。

从计算效率来看，CoFARS 模型将上下文偏好相似度计算和时序图聚合放在前置计算过程中。对于历史上下文数量为 $|\mathcal{C}|$、原型数量为 $|\mathcal{O}|$ 的用户，其上下文与原型之间的相似度计算复杂度为 $O(|\mathcal{C}||\mathcal{O}|)$，图聚合的主要计算复杂度为 $O(nL_g)$，其中 n 为上下文序列长度，L_g 为图聚合层数。在线上推理阶段，只需要利用最近 r 次行为计算短期兴趣，再与固定数量的偏好原型进行匹配，其原型筛选阶段的计算复杂度为 $O(r+|\mathcal{O}|)$。由于这一过程与候选物品数量无关，相比传统针对每个候选物品遍历完整历史序列的 $O(Bn)$ 检索方式，能够显著降低长序列筛选的计算开销。

总体来看，CoFARS 模型的核心贡献在于将用户历史行为序列中的上下文信息作为建模对象，通过偏好分布和原型学习识别相似的行为场景，再利用时序图将上下文转移关系引入长期兴趣表示。与 DisenCTR 模型直接在用户—物品交互图上进行多兴趣解耦不同，CoFARS 模型通过上下文偏好组织用户历史行为，从而在保留时间关联的同时实现与候选物品无关的子序列筛选。这种方法尤其适合用户行为受到时间、地点和场景等外部因素显著影响的推荐业务，例如外卖、到店和本地生活服务推荐。

20.8 本章小结

本章围绕推荐系统中的用户兴趣与行为序列建模展开，重点讨论了如何从用户不断累积的历史交互中提取稳定兴趣、短期兴趣和目标感知兴趣，并将这些兴趣表示用于粗排、精排以及召回任务。用户行为序列通常具有

长度长、兴趣多样、行为反馈异质、兴趣随时间变化以及在线计算预算严格等特点。因此，序列建模方法不仅需要提高表示能力，还需要在历史信息利用范围、目标相关性、计算复杂度和工程部署成本之间进行权衡。

首先，本章介绍了基于对比学习的行为序列建模方法。InfoNCE 将表示学习转化为正负样本之间的分类问题，通过密度比估计学习样本与上下文之间的关联，并给出了与互信息下界的联系。在此基础上，SimCLR 模型通过同一样本的不同增强视图构造正样本，MoCo 模型利用动量编码器和队列维护更多负样本，SimSiam 则通过停止梯度和预测头降低对负样本的依赖。面向推荐场景，本章进一步讨论了如何根据点击、长播、短播和主动负反馈等不同类型的行为构造正负关系，以及如何通过 Shuffle 等序列增广方式形成辅助对比目标。对比学习可以为行为 Embedding 提供额外监督，但其效果依赖正负样本语义是否可靠，错误的行为归因或过于容易的负样本都可能削弱训练信号。

随后，本章介绍了以 Attention 为核心的用户行为序列建模方法。DIN 模型通过候选物品与历史行为之间的注意力权重提取目标相关兴趣，DIEN 模型进一步利用兴趣演化网络刻画兴趣随行为序列的变化过程，DSIN 模型从 Session 粒度组织行为并建模 Session 内部和 Session 之间的兴趣演化，BST 模型则使用 Transformer 结构捕捉行为之间的全局依赖。TIN 模型进一步引入时间信息，增强模型对行为新鲜度和时间间隔的表达能力。这类方法具有较强的目标感知能力，但标准 Target Attention 通常需要对每个候选物品遍历完整历史序列，候选规模和序列长度增大后，计算量与内存访问开销会快速上升。

针对超长行为序列和在线资源约束，本章接着介绍了基于 Memory、召回检索和哈希的建模方法。MIMN 模型通过多级记忆网络持续更新用户历史兴趣，LMN 模型通过记忆机制处理更长的行为上下文；SIM、TWIN、MIRRN、FIM 和 LCN 等方法则先从海量历史中检索与当前目标相关的行为，再交由后续网络进行精细建模，从而将完整序列上的计算转化为较短子序列上的计算。Reformer 模型通过局部敏感哈希降低注意力计算规模，ETA 模型利用 SimHash 快速检索目标相关行为，ETA-Net 模型进一步面向工业部署优化长序列网络，SDIM 模型则通过哈希和兴趣聚合减少重复的目标感知计算。上述方法的共同思路是先压缩或筛选历史信息，再进行高表达能力的序列建模；实际应用中需要同时关注检索召回率、候选截断策略、哈希碰撞、索引更新和线上延迟。

本章还讨论了多模态信息对用户兴趣建模的补充作用。多模态 SIM、PSRQ、MISS 和 MUSE 等方法将图像、视频、文本以及物品结构信息引入用户行为表示，使模型能够利用用户消费内容中的语义和视觉信号，缓解仅依赖离散物品 ID 时的泛化不足。多模态特征通常具有维度高、生成成本高和版本更新复杂等特点，因此需要结合离线内容理解、Embedding 缓存、模态融合和在线服务能力进行设计。多模态信息应当服务于行为语义和兴趣表达，不能简单替代用户真实的交互反馈。

最后，本章介绍了基于时序图的用户兴趣建模方法。DisenCTR 模型通过图结构和兴趣解耦学习多个相对独立的用户兴趣，CoFARS 模型则将历史行为对应的上下文组织为时序图，通过偏好原型、上下文匹配和时间转移关系筛选与当前场景相关的行为子序列。与直接在用户—物品交互序列上进行注意力计算相比，时序图方法可以显式利用上下文之间的关联和转移，但同时需要维护图结构、原型表示以及相关的离线和在线更新链路。

20.9 参考文献

[1] Yue Cao et al. “Sampling is all you need on modeling long-term user behaviors for CTR prediction”. In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2022, pp. 2974–2983.

[2] Zheng Chai et al. “Longer: Scaling up long sequence modeling in industrial recommenders”. In: *Proceedings of the Nineteenth ACM Conference on Recommender Systems*. 2025, pp. 247–256.

[3] Jianxin Chang et al. “TWIN: TWo-stage interest network for lifelong user behavior modeling in CTR prediction at kuaishou”. In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2023, pp. 3785–3794.

[4] Qiwei Chen et al. “Behavior sequence transformer for e-commerce recommendation in alibaba”. In: *Proceedings of the 1st international workshop on deep learning practice for high-dimensional sparse data*. 2019, pp. 1–4.

[5] Qiwei Chen et al. “Efficient long sequential user data modeling for click-through rate prediction”. In: *arXiv preprint arXiv:2209.12212* (2022).

[6] Qiwei Chen et al. “End-to-end user behavior retrieval in click-through rateprediction model”. In: *arXiv preprint arXiv:2108.04468* (2021).

[7] Ting Chen et al. “A simple framework for contrastive learning of visual representations”. In: *International conference on machine learning*. PmLR. 2020, pp. 1597–1607.

[8] Xinlei Chen and Kaiming He. “Exploring simple siamese representation learning”. In: *2021 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*. IEEE. 2021, pp. 15745–15753.

[9] Yufei Feng et al. “Deep session interest network for click-through rate prediction”. In: *arXiv preprint arXiv:1905.06482* (2019).

[10] Zhichao Feng et al. “Context-based fast recommendation strategy for long user behavior sequence in Meituan Waimai”. In: *Companion Proceedings of the ACM Web Conference 2024*. 2024, pp. 355–363.

[11] Chengcheng Guo et al. “MISS: Multi-modal tree indexing and searching with lifelong sequential behavior for retrieval recommendation”. In: *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. 2025, pp. 5683–5690.

[12] Ting Guo et al. “Context-Aware Lifelong Sequential Modeling for Online Click-Through Rate Prediction”. In: *arXiv preprint arXiv:2502.12634* (2025).

[13] Kaiming He et al. “Momentum contrast for unsupervised visual representation learning”. In: *2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*. IEEE. 2020, pp. 9726–9735.

[14] Ruijie Hou et al. “Cross-domain lifelong sequential modeling for online click-through rate prediction”. In: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2024, pp. 5116–5125.

[15] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. “Reformer: The efficient transformer”. In: *arXiv preprint arXiv:2001.04451* (2020).

[16] Hongzun Liu et al. “De-Anchor: Mitigating Attention Polarization for Lifelong User Behavior Modeling in Click-Through Rate Prediction”. In: *Companion Proceedings of the ACM Web Conference 2024*. 2024, pp. 694–697.

[17] Hui Lu et al. “Large memory network for recommendation”. In: *Companion Proceedings of the ACM on Web Conference 2025*. 2025, pp. 1162–1166.

[18] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. “Representation learning with contrastive predictive coding”. In: *arXiv preprint arXiv:1807.03748* (2018).

[19] Qi Pi et al. “Practice on long sequential user behavior modeling for click-through rate prediction”. In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 2671–2679.

[20] Qi Pi et al. “Search-based user interest modeling with lifelong sequential behavior data for click-through rate prediction”. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2020, pp. 2685–2692.

[21] Zihua Si et al. “Twin v2: Scaling ultra-long user behavior sequence modeling for enhanced ctr prediction at kuaishou”. In: *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 2024, pp. 4890–4897.

[22] Guoquan Wang et al. “FIM: Frequency-aware multi-view interest modeling for local-life service recommendation”. In: *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2025, pp. 1748–1757.

[23] Shijia Wang et al. “Progressive semantic residual quantization for multimodal-joint interest modeling in music recommendation”. In: *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. 2025, pp. 6119–6127.

[24] Yifan Wang et al. “DisenCTR: Dynamic graph-based disentangled representation for click-through rate prediction”. In: *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 2022, pp. 2314–2318.

[25] Bin Wu et al. “MUSE: A simple yet effective multimodal search-based framework for lifelong user interest modeling”. In: *arXiv preprint arXiv:2512.07216* (2025).

[26] Songpei Xu et al. “Climber: Toward efficient scaling laws for large recommendation models”. In: *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. 2025, pp. 6193–6200.

[27] Xiang Xu et al. “Multi-granularity interest retrieval and refinement network for long-term user behavior modeling in ctr prediction”. In: *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*. 2025, pp. 2745–2755.

[28] Guorui Zhou et al. “Deep interest evolution network for click-through rate prediction”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 5941–5948.

[29] Guorui Zhou et al. “Deep interest network for click-through rate prediction”. In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, pp. 1059–1068.

[30] Haolin Zhou et al. “Temporal interest network for user response prediction”. In: *Companion Proceedings of the ACM Web Conference 2024*. 2024, pp. 413–422.