

推荐系统

概要

召回

基于物品的协同过滤(ItemCF)

基本思想

如果用户喜欢物品 $item_1$, 而且物品 $item_1$ 与 $item_2$ 相似, 那么用户很可能喜欢物品 $item_2$ 。

ItemCF 的实现

用户对物品的兴趣: $like(user, item_j)$

物品之间的相似度: $sim(item_j, item)$

预估用户对候选物品的兴趣: $\sum_j like(user, item_j) \times sim(item_j, item)$

物品的相似度

如果两个物品的受众重合度较高, 就判定为两个物品相似。

计算物品相似度(考虑用户喜欢的程度)

喜欢物品 i_1 的用户记作集合 \mathcal{W}_1

喜欢物品 i_2 的用户记作集合 \mathcal{W}_2

定义交集 $\mathcal{V} = \mathcal{W}_1 \cap \mathcal{W}_2$

两个物品的相似度:

$$sim(i_1, i_2) = \frac{|\mathcal{V}|}{\sqrt{|\mathcal{W}_1| \cdot |\mathcal{W}_2|}}.$$

计算物品相似度(考虑用户喜欢的程度)

喜欢物品 i_1 的用户记作集合 \mathcal{W}_1

喜欢物品 i_2 的用户记作集合 \mathcal{W}_2

定义交集 $\mathcal{V} = \mathcal{W}_1 \cap \mathcal{W}_2$

两个物品的相似度:

$$sim(i_1, i_2) = \frac{\sum_{v \in \mathcal{V}} like(v, i_1) \cdot like(v, i_2)}{\sqrt{\sum_{u_1 \in \mathcal{W}_1} like^2(u_1, i_1)} \cdot \sqrt{\sum_{u_2 \in \mathcal{W}_2} like^2(u_2, i_2)}} \text{余弦相似度 (cosine similarity)}$$

ItemCF 召回的完整流程

事先做离线计算

建立“用户 → 物品”的索引

- 记录每个用户最近点击、交互过的物品ID。
- 给定任意用户ID, 可以找到他近期感兴趣的物品列表。

建立“物品 → 物品”的索引

- 计算物品之间两两相似度。
- 对于每个物品，索引它最相似的 k 个物品。
- 给定任意物品ID，可以快速找到它最相似的 k 个物品。

线上做召回

1. 给定用户ID，通过“用户 → 物品”索引，找到用户近期感兴趣的物品列表 (last-n)。
2. 对于 last-n 列表中每个物品，通过“物品 → 物品”的索引，找到 top-k 相似物品。
3. 对于召回的相似物品（最多有 nk 个），用公式预估用户对物品的兴趣分数。
4. 返回分数最高的 100 个物品，作为推荐结果。

用索引，离线计算量大，线上计算量小。

总结

ItemCF的原理

用户喜欢物品 i_1 ，那么用户喜欢与物品 i_1 相似的物品 i_2 。

物品相似度：

- 如果喜欢 i_1 、 i_2 的用户有很大的重叠，那么 i_1 与 i_2 相似。
- 公式：

$$sim(i_1, i_2) = \frac{|\mathcal{W}_1 \cap \mathcal{W}_2|}{\sqrt{|\mathcal{W}_1| \cdot |\mathcal{W}_2|}}$$

ItemCF 召回通道

维持两个索引：

- 用户 → 物品列表：用户最近交互过的 n 个物品。
- 物品 → 物品列表：相似度最高的 k 个物品。

线上做召回：

- 利用两个索引，每次召回 nk 个物品。
- 预估用户对每个物品的兴趣分数：

$$\sum_j like(user, item_j) \times sim(item_j, item).$$

- 返回分数最高的 100 个物品，作为召回结果。

Swing召回通道

Swing 模型

用户 u_1 喜欢的物品记作集合 \mathcal{J}_1 。

用户 u_2 喜欢的物品记作集合 \mathcal{J}_2 。

定义两个用户的重合度：

$$\text{overlap}(u_1, u_2) = |\mathcal{J}_1 \cap \mathcal{J}_2|$$

用户 u_1 和 u_2 的重合度高，则他们可能来自一个小圈子，要降低他们的权重。

Swing 模型

喜欢物品 i_1 的用户记作集合 \mathcal{W}_1 。

喜欢物品 i_2 的用户记作集合 \mathcal{W}_2 。

定义交集 $\mathcal{V} = \mathcal{W}_1 \cap \mathcal{W}_2$ 。

两个物品的相似度：

$$sim(i_1, i_2) = \sum_{u_1 \in \mathcal{V}} \sum_{u_2 \in \mathcal{V}} \frac{1}{\alpha + overlap(u_1, u_2)}$$

总结

- Swing 与 ItemCF 唯一的区别在于物品相似度。
- **ItemCF**: 两个物品重合的用户比例高，则判定两个物品相似。
- **Swing**: 额外考虑重合的用户是否来自一个小圈子。
 - 同时喜欢两个物品的用户记作集合 \mathcal{V} 。
 - 对于 \mathcal{V} 中的用户 u_1 和 u_2 , 重合度记作 $overlap(u_1, u_2)$ 。
 - 两个用户重合度大，则可能来自一个小圈子，权重降低。

基于用户的协同过滤 (UserCF)

基本思想

如果用户 $user_1$ 跟用户 $user_2$ 相似，而且 $user_2$ 喜欢某物品，那么用户 $user_1$ 也很可能喜欢该物品。

UserCF 的实现

用户之间的相似度： $sim(user, user_j)$

用户对物品的兴趣： $like(user_j, item)$

预估用户对候选物品的兴趣： $\sum_j sim(user, user_j) \times like(user_j, item)$

用户的相似度

计算用户相似度

用户 u_1 喜欢的物品记作集合 \mathcal{J}_1 。

用户 u_2 喜欢的物品记作集合 \mathcal{J}_2 。

定义交集 $I = \mathcal{J}_1 \cap \mathcal{J}_2$ 。

两个用户的相似度：

$$sim(u_1, u_2) = \frac{|I|}{\sqrt{|\mathcal{J}_1| \cdot |\mathcal{J}_2|}}$$

降低热门物品权重

用户 u_1 喜欢的物品记作集合 \mathcal{J}_1 。

用户 u_2 喜欢的物品记作集合 \mathcal{J}_2 。

定义交集 $I = \mathcal{J}_1 \cap \mathcal{J}_2$ 。

两个用户的相似度：

$$sim(u_1, u_2) = \frac{\sum_{l \in I} \frac{1}{\log(1+n_l)}}{\sqrt{|\mathcal{J}_1| \cdot |\mathcal{J}_2|}}.$$

其中， n_l 表示喜欢物品 l 的用户数量，反映物品的热门程度。

UserCF 召回的完整流程

事先做离线计算

建立“用户 → 物品”的索引

- 记录每个用户最近点击、交互过的物品ID。
- 给定任意用户ID，可以找到他近期感兴趣的物品列表。

建立“用户 → 用户”的索引

- 对于每个用户，索引他最相似的 k 个用户。
- 给定任意用户ID，可以快速找到他最相似的 k 个用户。

线上做召回

1. 给定用户ID，通过“用户 → 用户”索引，找到 top-k 相似用户。
2. 对于每个 top-k 相似用户，通过“用户 → 物品”索引，找到用户近期感兴趣的物品列表 (last-n)。
3. 对于召回的 nk 个相似物品，用公式预估用户对每个物品的兴趣分数。
4. 返回分数最高的 100 个物品，作为召回结果。

总结

UserCF 的原理

用户 u_1 跟用户 u_2 相似，而且 u_2 喜欢某物品，那么 u_1 也可能喜欢该物品。

用户相似度：

- 如果用户 u_1 和 u_2 喜欢的物品有很大的重叠，那么 u_1 和 u_2 相似。
- 公式：

$$sim(u_1, u_2) = \frac{|\mathcal{J}_1 \cap \mathcal{J}_2|}{\sqrt{|\mathcal{J}_1| \cdot |\mathcal{J}_2|}}.$$

UserCF 召回通道

维持两个索引：

- 用户 → 物品列表：用户近期交互过的 n 个物品。
- 用户 → 用户列表：相似度最高的 k 个用户。

线上做召回：

- 利用两个索引，每次召回 nk 个物品。

- 预估用户 $user$ 对每个物品 $item$ 的兴趣分数：

$$\sum_j sim(user, user_j) \times like(user_j, item)。$$

- 返回分数最高的 100 个物品，作为召回结果。

离散特征处理

1. **建立字典**：把类别映射成序号。

- 中国 → 1
- 美国 → 2
- 印度 → 3

2. **向量化**：把序号映射成向量。

- One-hot 编码：把序号映射成高维稀疏向量。
- Embedding：把序号映射成低维稠密向量。

独热编码(one-hot编码)

独热编码表示国籍特征

国籍：中国、美国、印度等 200 种类别。

字典：中国 → 1, 美国 → 2, 印度 → 3, ...

One-hot 编码：用 200 维稀疏向量表示国籍。

- 未知 → 0 → [0,0,0,0,...,0]
- 中国 → 1 → [1,0,0,0,...,0]
- 美国 → 2 → [0,1,0,0,...,0]
- 印度 → 3 → [0,0,1,0,...,0]

Embedding (嵌入)

可以将独热编码映射为嵌入向量

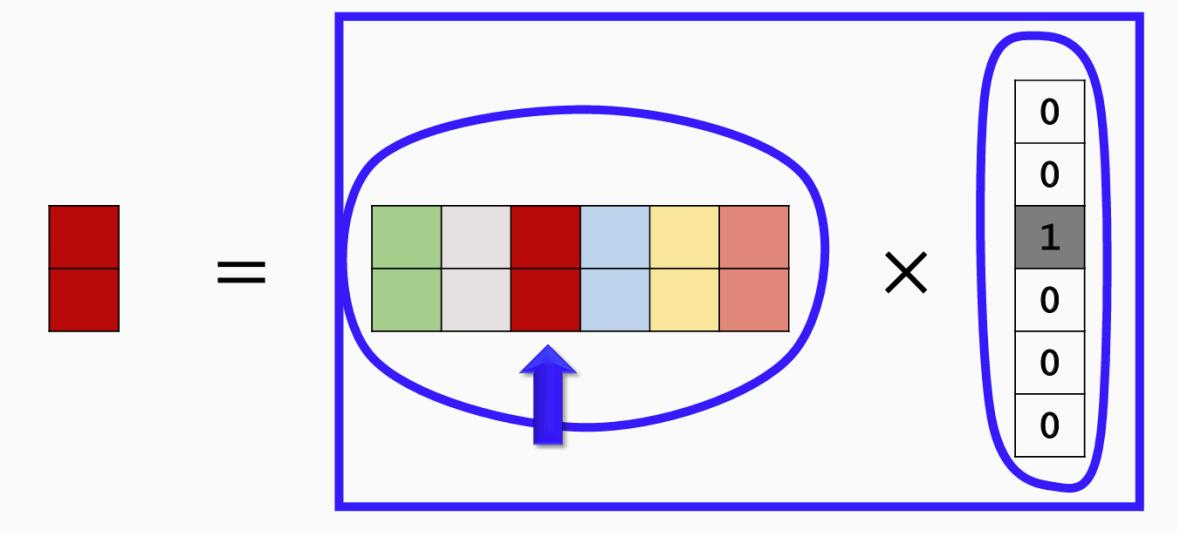
参数数量：向量维度 × 类别数量。

- 设 embedding 得到的向量都是 4 维的。
- 一共有 200 个国籍。
- 参数数量 = $4 \times 200 = 800$ 。

编程实现：TensorFlow、PyTorch 提供 embedding 层。

- 参数以矩阵的形式保存，矩阵大小是 向量维度 × 类别数量。
- 输入是序号，比如“美国”的序号是 2。
- 输出是向量，比如“美国”对应参数矩阵的第 2 列。

$$\text{Embedding} = \text{参数矩阵} \times \text{One-Hot 向量}$$



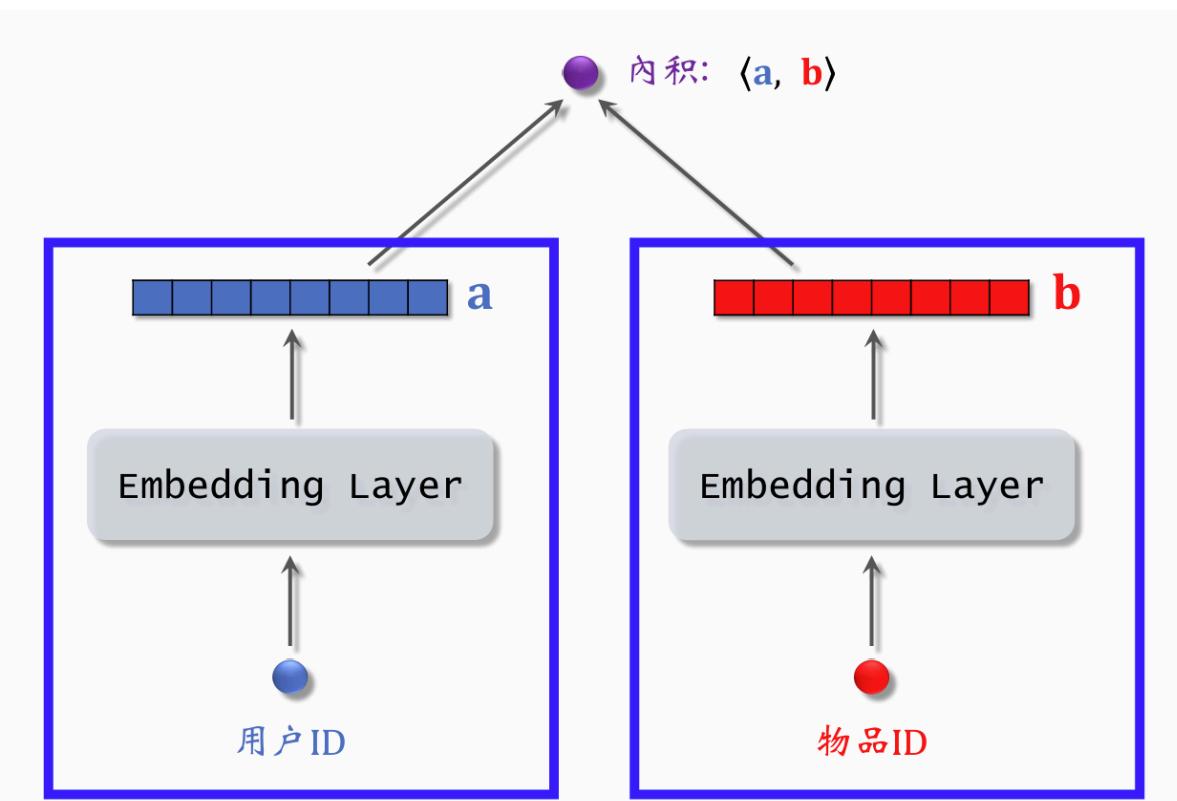
总结

离散特征处理：one-hot 编码、embedding。

类别数量很大时，用 embedding。

- Word embedding。
- 用户 ID embedding。
- 物品 ID embedding。

矩阵补充



embedding 层矩阵 A 输出的向量 a 是矩阵的一列，用户的数量等于矩阵的列数

embedding 层矩阵 B 输出的向量 b 是矩阵的一列，物品的数量等于矩阵的列数

基本想法

用户 embedding 参数矩阵记作 A 。第 u 号用户对应矩阵第 u 列，记作向量 a_u

物品 embedding 参数矩阵记作 B 。第 i 号物品对应矩阵第 i 列，记作向量 b_i

内积 $\langle a_u, b_i \rangle$ 是第 u 号用户对第 i 号物品兴趣的预估值

训练模型的目的是学习矩阵 A 和 B ，使得预估值拟合真实观测的兴趣分数，矩阵 A 和 B 是 embedding 层的参数

数据集

数据集：(用户ID, 物品ID, 兴趣分数) 的集合，记作

$$\Omega = \{(u, i, y)\}.$$

数据集中的兴趣分数是系统记录的，例如：

- 曝光但是没有点击 $\rightarrow 0$ 分
- 点击、点赞、收藏、转发 \rightarrow 各算 1 分
- 分数最低是 0，最高是 4

训练

把用户 ID、物品 ID 映射成向量。

- 第 u 号用户 \rightarrow 向量 a_u
- 第 i 号物品 \rightarrow 向量 b_i

求解优化问题，得到参数 A 和 B ，可采用梯度下降

$$\min_{A, B} \sum_{(u, i, y) \in \Omega} (y - \langle a_u, b_i \rangle)^2.$$

矩阵补充

矩阵补充																	
绿色位置表示曝光给用户的物品；灰色位置表示没有曝光。																	
第3号用户 对 第2号物品 的兴趣分数 等于4	1			0			1		0			0		0		0	2
	0	0	4	1			1	1	0	1	2	0	1	1		1	
	0			1			0		1		0	0		0	4		
	0	0	0	0			0		1		0	0		0		0	
	1			0			1		0		0			0		0	
	1	0		0			4		0		0		0		3		
	2	0	2		1		1		1		0		0	0	1	0	
	0	2		1			1		4		2		0		1	0	
	3																

模型训练后可以将灰色位置补全，这时再根据兴趣分数做推荐

在实践中效果不好.....

缺点1：仅用 ID embedding，没利用物品、用户属性。

- 物品属性：类别、关键词、地理位置、作者信息。
- 用户属性：性别、年龄、地理定位、感兴趣的类别。
- 双塔模型可以看做矩阵补充的升级版。

缺点2：负样本的选取方式不对。

- 样本：用户—物品的二元组，记作 (u, i) 。
- 正样本：曝光之后，有点击、交互。（正确的做法）
- 负样本：曝光之后，没有点击、交互。（错误的做法）

缺点3：做训练的方法不好。

- 内积 $\langle \mathbf{a}_u, \mathbf{b}_i \rangle$ 不如余弦相似度。
- 用平方损失（回归），不如用交叉熵损失（分类）。

模型存储

1. 训练得到矩阵 **A** 和 **B**。
 - **A** 的每一列对应一个用户。
 - **B** 的每一列对应一个物品。
2. 把矩阵 **A** 的列存储到 key-value 表。
 - key 是用户 ID，value 是 **A** 的一列。
 - 给定用户 ID，返回一个向量（用户的 *embedding*）。
3. 矩阵 **B** 的存储和索引比较复杂。

线上服务

1. 把用户 ID 作为 key，查询 key-value 表，得到该用户的向量，记作 **a**。
2. 最近邻查找：查找用户最有可能感兴趣的 k 个物品，作为召回结果。
 - 第 i 号物品的 *embedding* 向量记作 \mathbf{b}_i 。
 - 内积 $\langle \mathbf{a}, \mathbf{b}_i \rangle$ 是用户对第 i 号物品兴趣的预估。
 - 返回内积最大的 k 个物品。

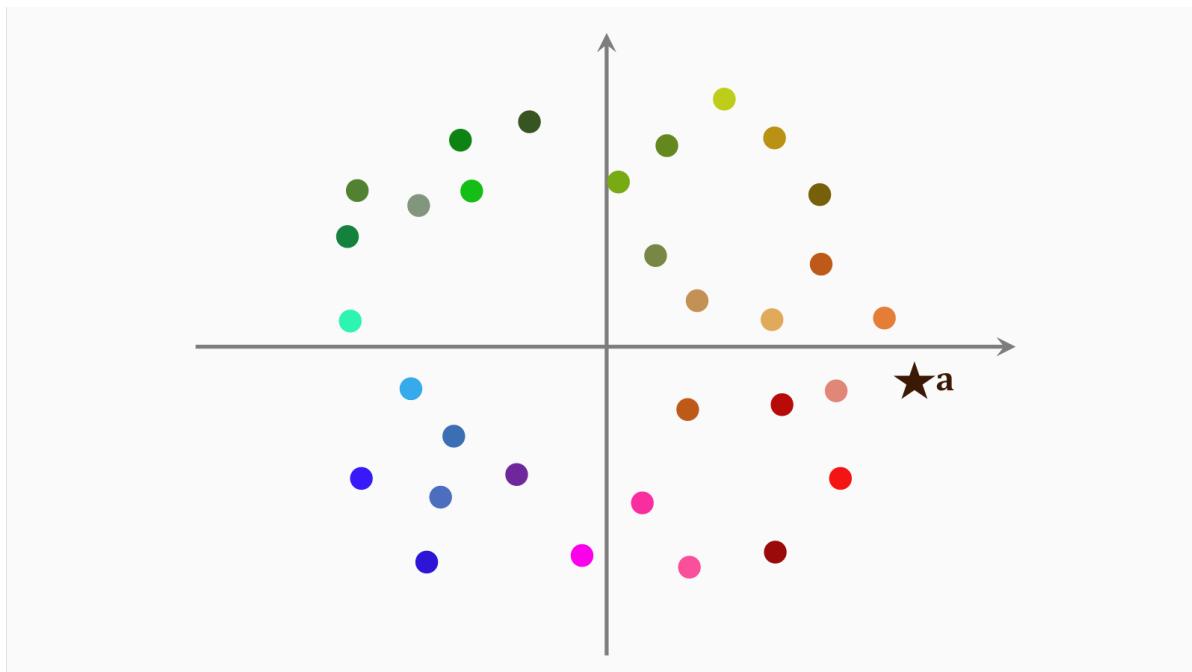
如果枚举所有物品，时间复杂度正比于物品数量。因为要找到 k 个最感兴趣的物品就要算出来用户对所有物品的兴趣分数

支持最近邻查找的系统

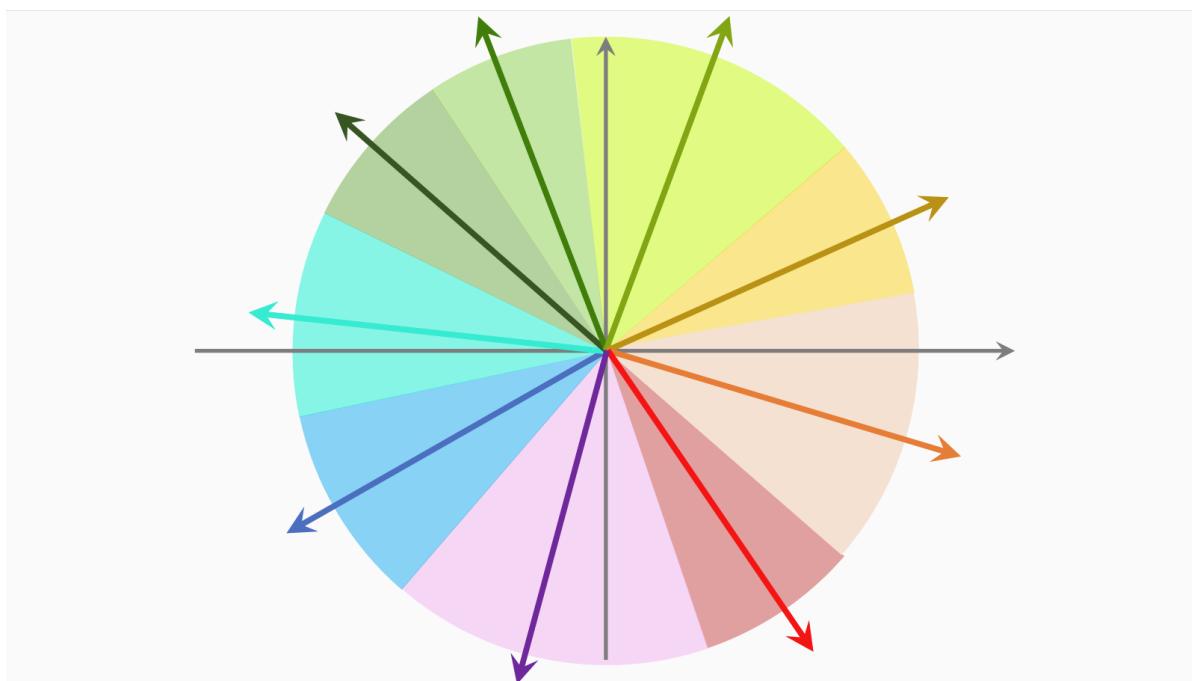
系统：Milvus、Faiss、HnswLib、等等。

衡量最近邻的标准：

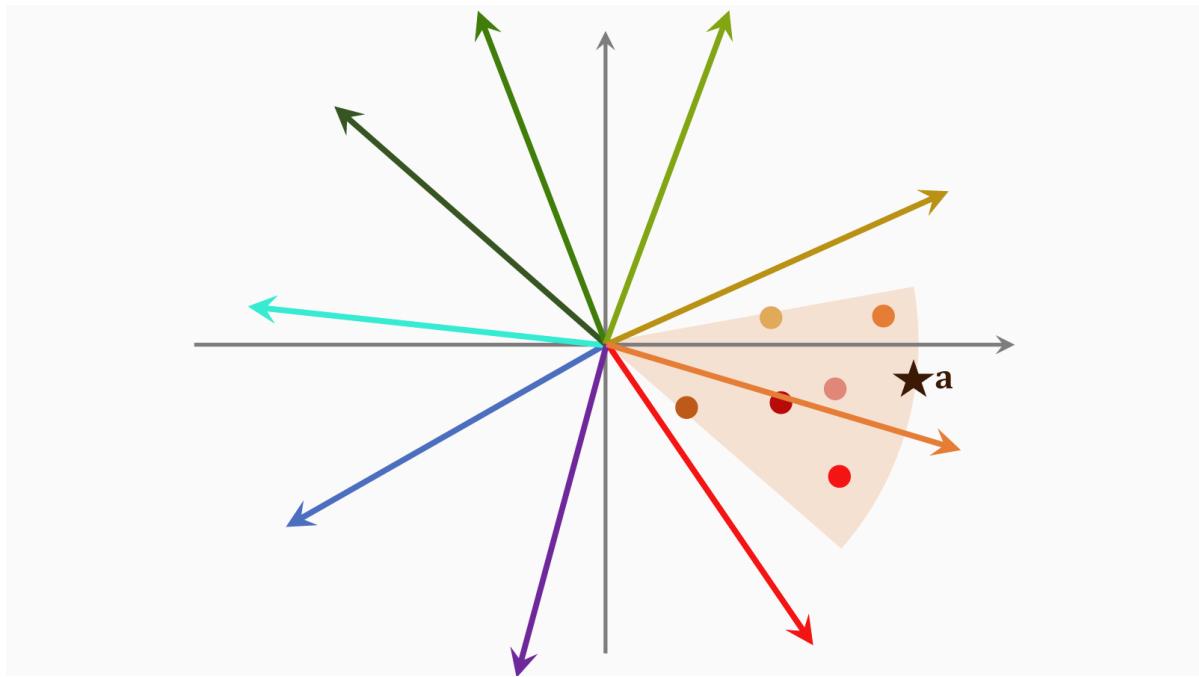
- 欧式距离最小（L2 距离）
- 向量内积最大（内积相似度）
- 向量夹角余弦最大（cosine 相似度）



a 代表某个用户的embedding向量，图中的散点代表物品的embedding向量。要找到与 a 最近邻的向量（最近邻在矩阵补充中就是向量内积最大）。



将图中的散点划分为若干区域，每一个区域用一个向量来代替。也就是用一个区域向量来代替所在区域内的若干物品向量。



计算与 a 最近邻的区域向量，找到后再计算出这个区域内与 a 最近邻的 k 个物品向量。

总结

矩阵补充

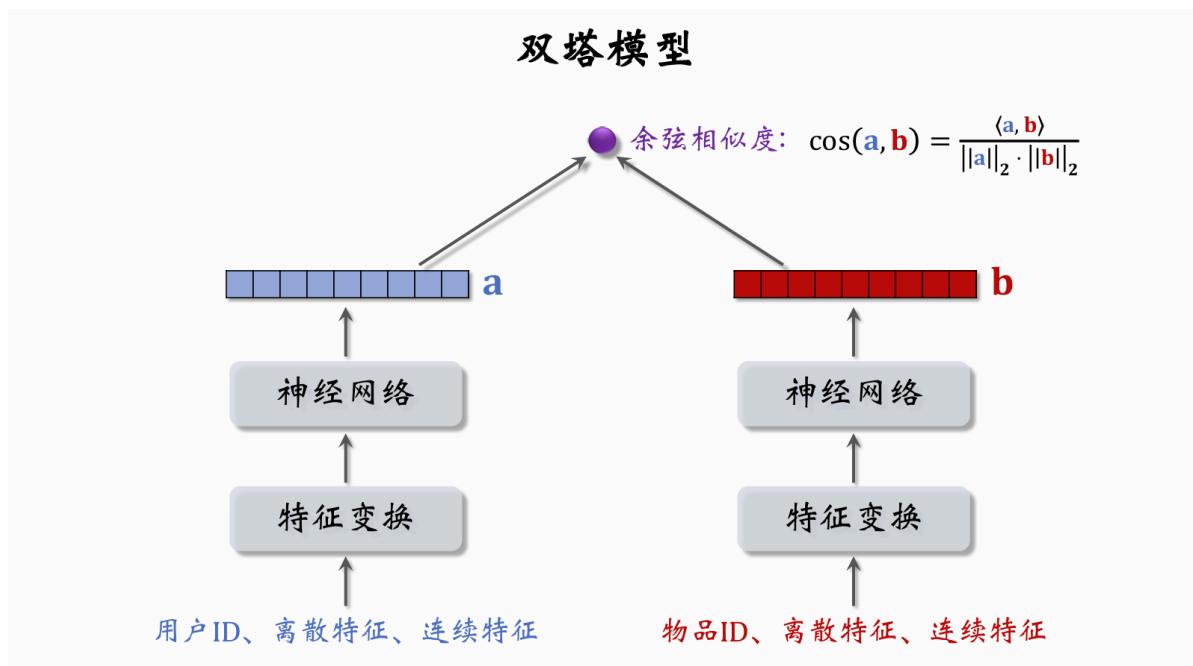
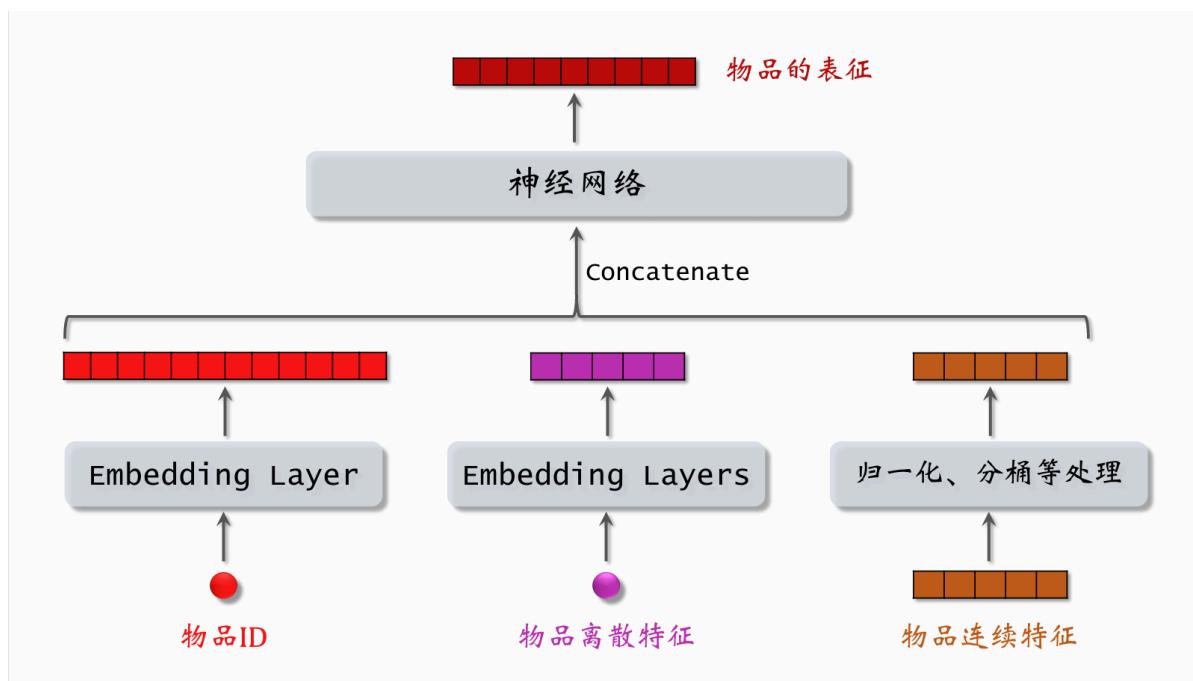
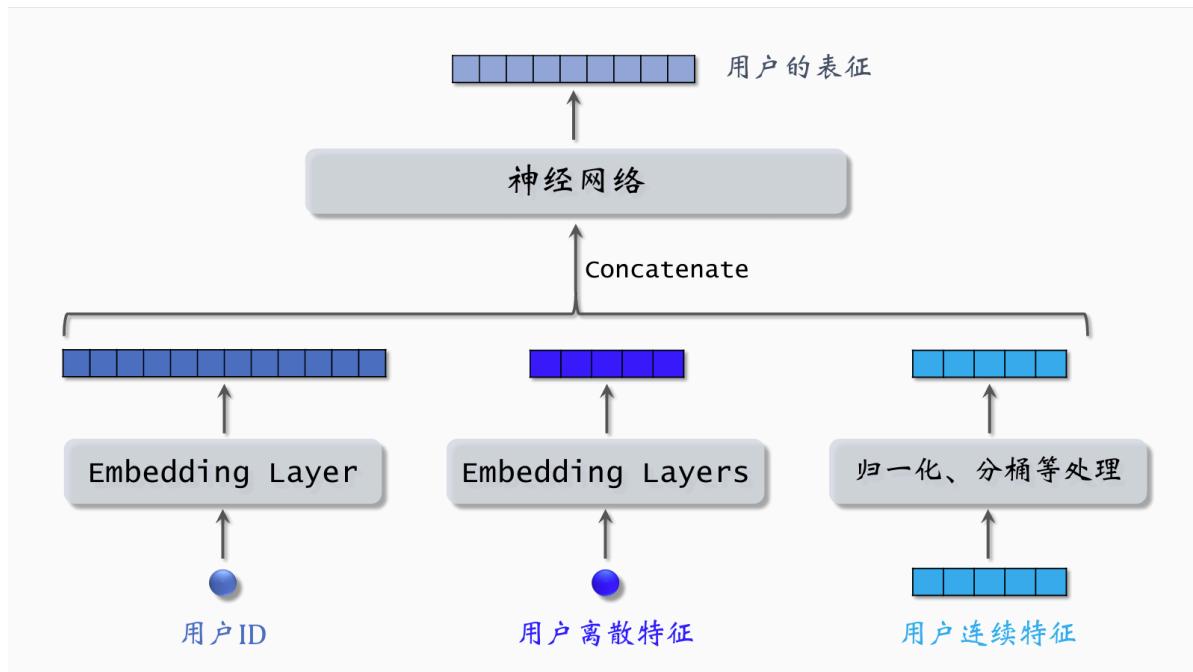
- 把物品 ID、用户 ID 做 embedding，映射成向量。
- 两个向量的内积 $\langle \mathbf{a}_u, \mathbf{b}_i \rangle$ 作为用户 u 对物品 i 兴趣的预估。
- 让 $\langle \mathbf{a}_u, \mathbf{b}_i \rangle$ 拟合真实观测的兴趣分数，学习模型的 embedding 层参数。
- 矩阵补充模型有很多缺点，效果不好。

线上召回

- 把用户向量 \mathbf{a} 作为 query，查找使得 $\langle \mathbf{a}, \mathbf{b}_i \rangle$ 最大化的物品 i 。
- 暴力枚举速度太慢。实践中用近似最近邻查找。
- Milvus、Faiss、HnswLib 等向量数据库支持近似最近邻查找。

双塔模型：模型和训练

双塔模型



双塔模型的训练

- Pointwise: 独立看待每个正样本、负样本，做简单的二元分类。
- Pairwise: 每次取一个正样本、一个负样本。
- Listwise: 每次取一个正样本、多个负样本。

正负样本的选择

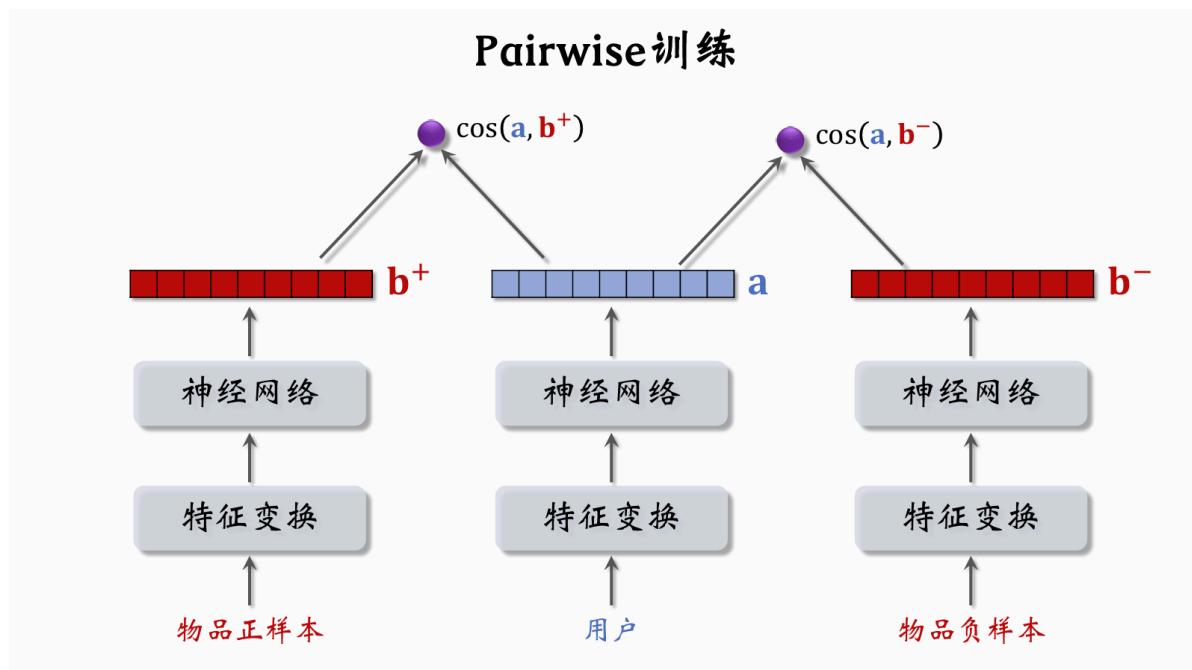
- 正样本: 用户点击的物品。
- 负样本 [1, 2]:
 - 没有被召回的?
 - 召回但是被粗排、精排淘汰的?
 - 曝光但是未点击的?

Pointwise训练

- 把召回看做二元分类任务。
- 对于正样本，鼓励 $\cos(a, b^+)$ 接近 +1。
- 对于负样本，鼓励 $\cos(a, b^-)$ 接近 -1。
- 控制正负样本数量为 1 : 2 或者 1 : 3。

Pairwise训练

两个物品塔的参数是相同的。



基本想法: 鼓励 $\cos(a, b^+)$ 大于 $\cos(a, b^-)$ 。

- 如果 $\cos(a, b^+)$ 大于 $\cos(a, b^-) + m$, 则没有损失。其中 m 为超参数, 需要自己设置。
- 否则, 损失等于 $\cos(a, b^-) + m - \cos(a, b^+)$ 。

损失函数

Triplet hinge loss:

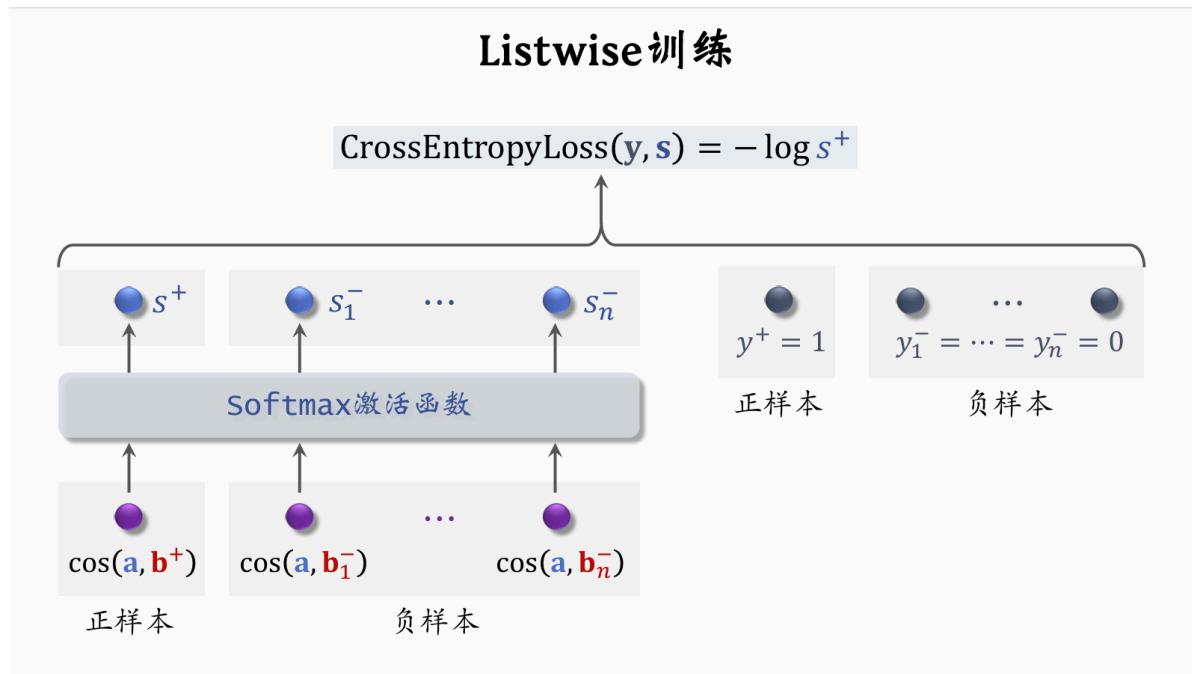
$$L(a, b^+, b^-) = \max\{0, \cos(a, b^-) + m - \cos(a, b^+)\}$$

Triplet logistic loss:

$$L(\mathbf{a}, \mathbf{b}^+, \mathbf{b}^-) = \log(1 + \exp[\sigma \cdot (\cos(\mathbf{a}, \mathbf{b}^-) - \cos(\mathbf{a}, \mathbf{b}^+))]).$$

Listwise 训练

- 一条数据包含：
 - 一个用户，特征向量记作 \mathbf{a} 。
 - 一个正样本，特征向量记作 \mathbf{b}^+ 。
 - 多个负样本，特征向量记作 $\mathbf{b}_1^-, \dots, \mathbf{b}_n^-$ 。
- 鼓励 $\cos(\mathbf{a}, \mathbf{b}^+)$ 尽量大。
- 鼓励 $\cos(\mathbf{a}, \mathbf{b}_1^-), \dots, \cos(\mathbf{a}, \mathbf{b}_n^-)$ 尽量小。



鼓励 $\cos(\mathbf{a}, \mathbf{b}^+)$ 尽量接近于1，鼓励 $\cos(\mathbf{a}, \mathbf{b}_1^-), \dots, \cos(\mathbf{a}, \mathbf{b}_n^-)$ 尽量接近于0

损失函数为交叉熵损失函数。

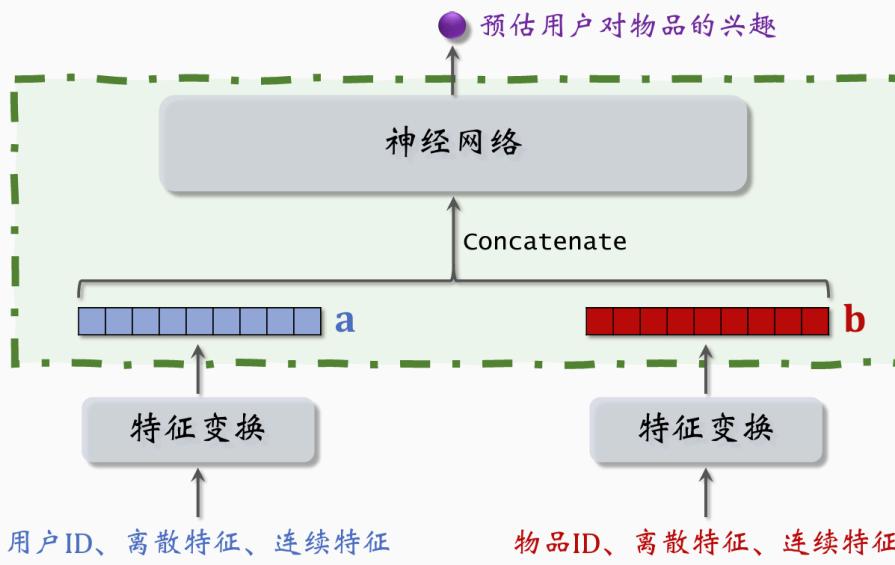
总结

双塔模型

- 用户塔、物品塔各输出一个向量。
- 两个向量的余弦相似度作为兴趣的预估值。
- 三种训练方式：
 - **Pointwise**: 每次用一个用户、一个物品（可正可负）。
 - **Pairwise**: 每次用一个用户、一个正样本、一个负样本。
 - **Listwise**: 每次用一个用户、一个正样本、多个负样本。

不适用于召回的模型：前期融合模型

不适用于召回的模型



采用双塔模型这种后期融合的召回方式，其优点在于先前就可以借助物品塔计算好所有物品的表示。之后每次来一个用户，将其通过用户塔可以得到他的表示，借助快速最近邻方法，可以快速召回k个和用户表示相近的物品。

而如果采用前期融合的方式，就无法预先计算好所有物品的表示，要召回k个物品，必须让用户的特征和每个物品的特征融合后输入神经网络得到一个感兴趣分数，这样必须把每个物品都计算一遍，发挥不了快速最近邻的方法的优势。

双塔模型：正负样本

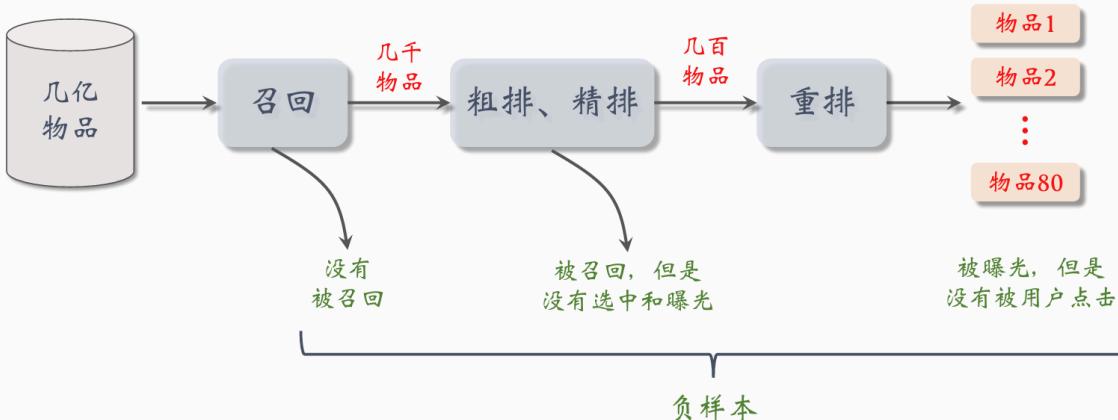
正样本

- 正样本：曝光而且有点击的用户—物品二元组。 (用户对物品感兴趣)
- 问题：少部分物品占据大部分点击，导致正样本 大多是热门物品
- 解决方案：过采样冷门物品，或降采样热门物品
- 过采样 (up-sampling) : 一个样本出现多次
- 降采样 (down-sampling) : 一些样本被抛弃

如何选择负样本

召回，粗排，精排和重排的负样本选择标准不同

如何选择负样本？



简单负样本

简单负样本：全体物品

- 未被召回的物品，大概是用户不感兴趣的。
- 未被召回的物品 \approx 整体物品
- 从整体物品中做抽样，作为负样本。
- 均匀抽样 or 非均匀抽样？

均匀抽样：对冷门物品不公平

- 正样本大多是热门物品。
- 如果均匀抽样产生负样本，负样本大多是冷门物品。

非均匀抽样：目的是打压热门物品

- 负样本抽样概率与热门程度（点击次数）正相关。
- 抽样概率 \propto (点击次数) $^{0.75}$ 。0.75是经验值。

简单负样本：Batch内负样本

- 一个 batch 内有 n 个正样本。
- 一个用户 p 和 $n - 1$ 个物品组成负样本。
- 这个 batch 内一共有 $n(n - 1)$ 个负样本。
- 都是简单负样本。（因为第一个用户不喜欢第二个物品。）
- 一个物品出现在 batch 内的概率 \propto 点击次数。物品越热门，出现在batch内的概率就越高。
- 物品成为负样本的概率本该是 \propto 点击次数 $^{0.75}$ ，但在这里是 \propto 点击次数。
- 热门物品成为负样本的概率过大。
- 物品 i 被抽样到的概率：
$$p_i \propto \text{点击次数}$$
 物品越热门，被抽样到的概率就越高。
- 预估用户对物品 i 的兴趣：
$$\cos(\mathbf{a}, \mathbf{b}_i)$$

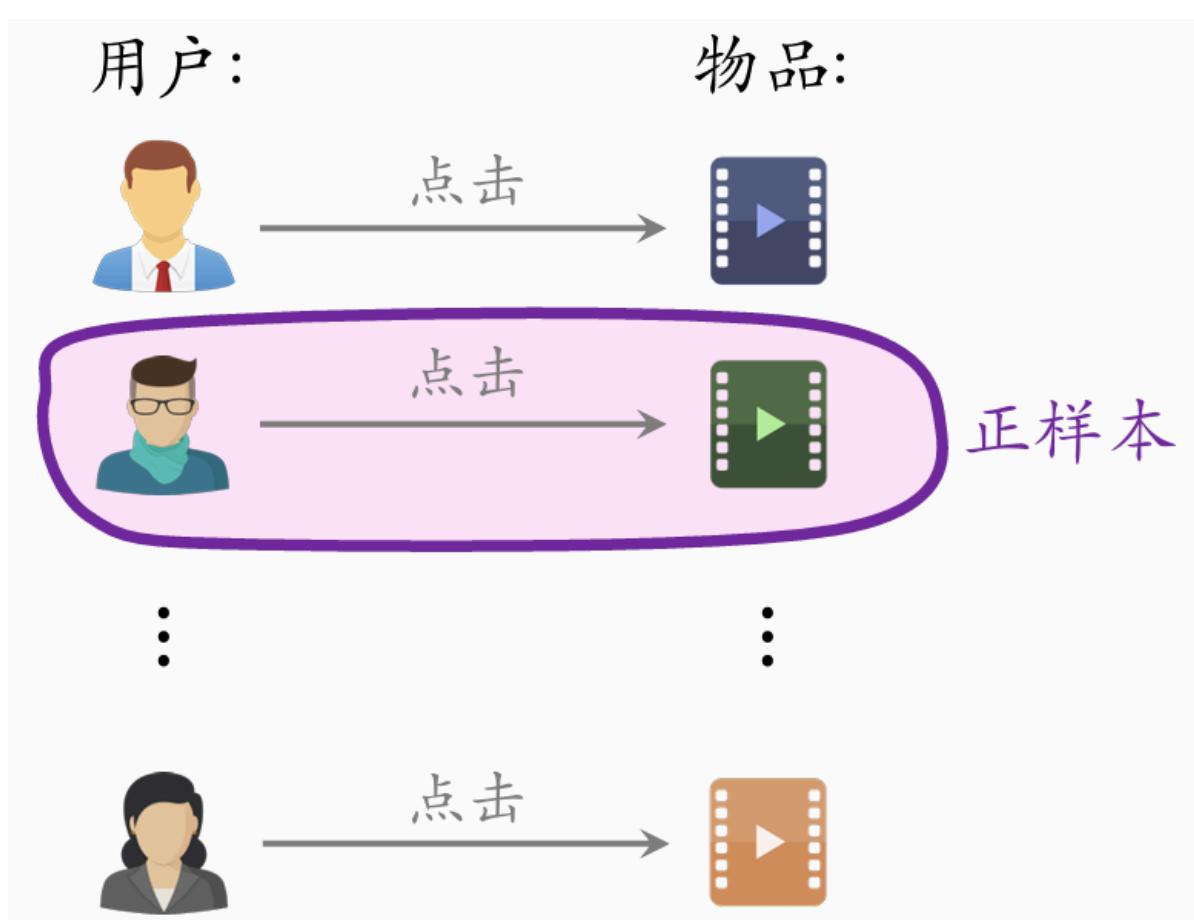
- 做训练的时候，调整为：

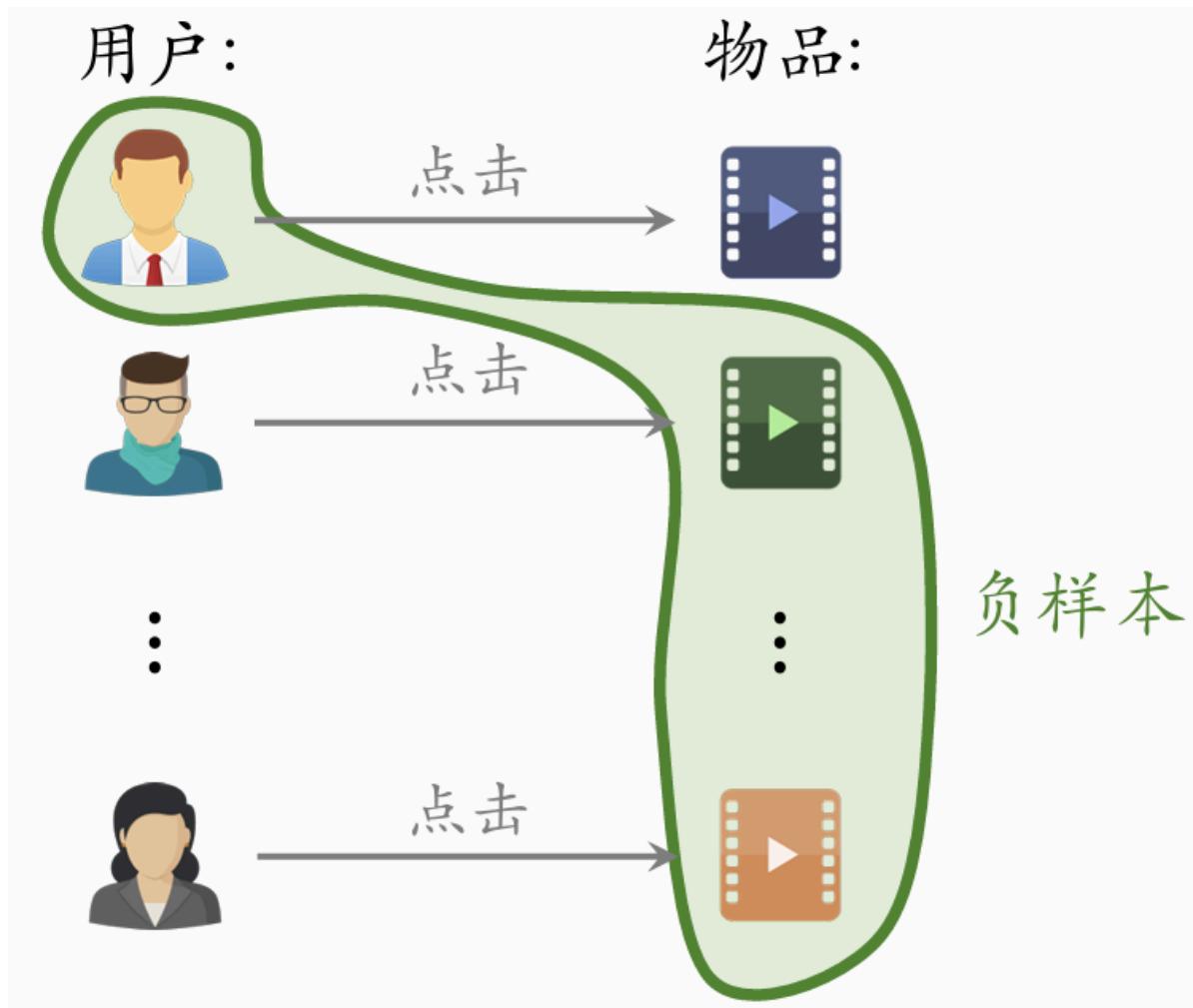
$$\cos(\mathbf{a}, \mathbf{b}_i) - \log p_i$$

这样可以纠偏，避免打压热门物品

具体原理： $\cos(\mathbf{a}, \mathbf{b}_i) - \log p_i$ 作为训练用，物品越热门，这个值越小。当某热门物品与某冷门物品的 $\cos(\mathbf{a}, \mathbf{b}_i)$ 相同时，因为热门物品的 $\cos(\mathbf{a}, \mathbf{b}_i) - \log p_i$ 更小，其训练所用的目标值就更小。

当模型训练完成后，输入相同的热门物品与冷门物品，热门物品的 $\cos(\mathbf{a}, \mathbf{b}_i)$ 就大于冷门物品。





困难负样本

困难负样本

- 被粗排淘汰的物品（比较困难）。
- 精排分数靠后的物品（非常困难）。

对正负样本做二元分类：

- 整体物品（简单）分类准确率高。
- 被粗排淘汰的物品（比较困难）容易分错。
- 精排分数靠后的物品（非常困难）更容易分错。

训练数据

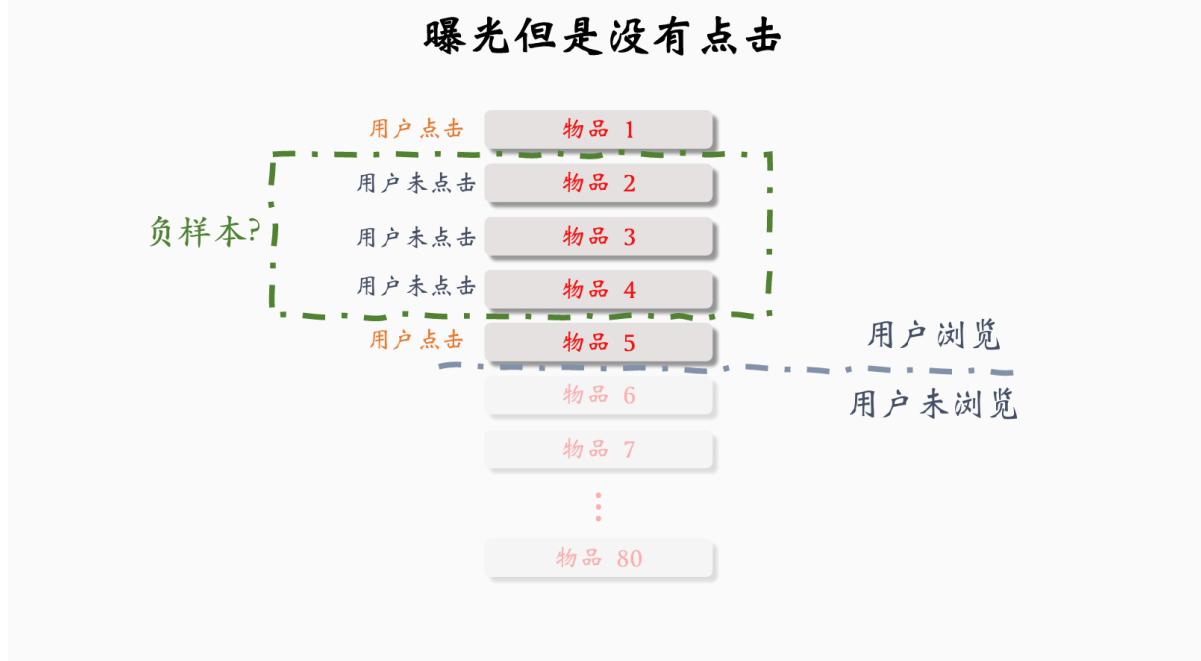
- 混合几种负样本。
- 50% 的负样本是整体物品（简单负样本）。
- 50% 的负样本是没通过排序的物品（困难负样本）。

常见的错误

训练找回模型不能用这类负样本

训练排序模型会用这类负样本

曝光但是没有点击



选择负样本的原理

召回的目标：快速找到用户可能感兴趣的物品。

- 整体物品 (easy)：绝大多数是用户根本不感兴趣的。
- 被排序淘汰 (hard)：用户可能感兴趣，但是不够感兴趣。
- 有曝光没点击 (没用)：用户感兴趣，可能碰巧没有点击。
 - 可以作为排序的负样本，不能作为召回的负样本。

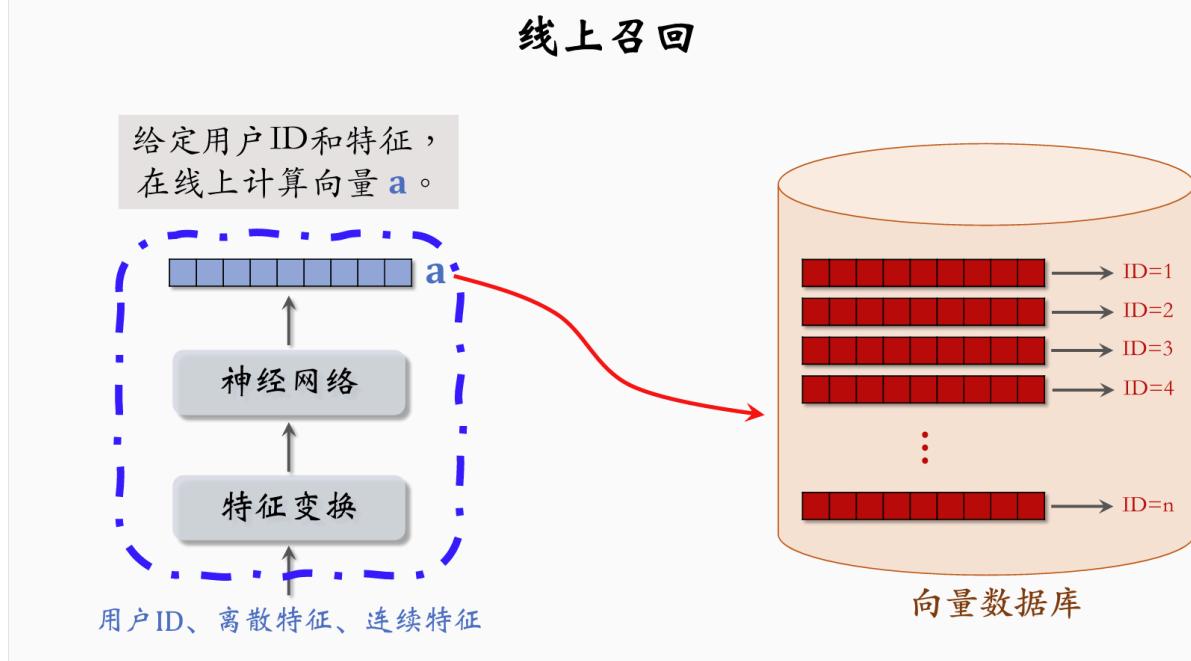
总结

- 正样本：曝光而且有点击。
- 简单负样本：
 - 整体物品。
 - batch 内负样本。
- 困难负样本：被召回，但是被排序淘汰。
- 错误：曝光，但是未点击的物品做召回的负样本。

双塔模型：线上召回和更新

线上召回

线上召回



双塔模型的召回

离线存储: 把物品向量 b 存入向量数据库。

1. 完成训练之后，用物品塔计算每个物品的特征向量 b 。
2. 把几亿个物品向量 b 存入向量数据库（比如 Milvus、Faiss、HnswLib）。
3. 向量数据库建索引，以便加速最近邻查找。

线上召回: 查找用户最感兴趣的 k 个物品。

1. 给定用户 ID 和画像，线上用神经网络算用户向量 a 。
2. 最近邻查找：
 - 把向量 a 作为 query，调用向量数据库做最近邻查找。
 - 返回余弦相似度最大的 k 个物品，作为召回结果。

事先存储物品向量 b ，线上现算用户向量 a ，why？

- 每做一次召回，用到一个用户向量 a ，几亿物品向量 b 。
(线上计算物品向量的代价过大。)
- 用户兴趣动态变化，而物品特征相对稳定。
(可以离线存储用户向量，但不利于推荐效果。)

模型更新

全量更新 vs 增量更新

全量更新: 今天凌晨，用昨天全天的数据训练模型。

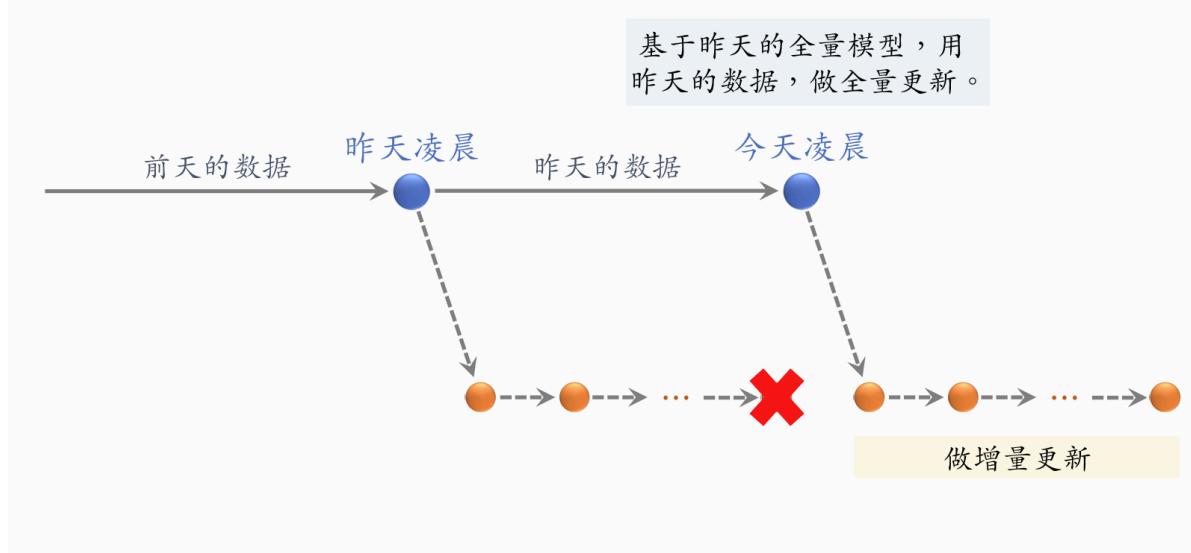
- 在昨天模型参数的基础上做训练。（不是随机初始化）
- 用昨天的数据，训练 1 epoch，即每天数据只用一遍。
- 发布新的 **用户塔神经网络** 和 **物品向量**，供线上召回使用。
- 全量更新对数据流、系统的要求比较低。

增量更新: 做 online learning 更新模型参数。

- 用户兴趣会随时发生变化。

- 实时收集线上数据，做流式处理，生成 TFRecord 文件。
- 对模型做 online learning，增量更新 ID Embedding 参数。
- 发布用户 ID Embedding，供用户塔在线上计算用户向量。

全量更新 vs 增量更新



问题：能否只做增量更新，不做全量更新？

- 一天中每一时段用户的行为不同，小时级数据有偏；分钟级数据偏差更大。
- 全量更新：random shuffle 一天的数据，做 1 epoch 训练。
- 增量更新：按照数据从早到晚的顺序，做 1 epoch 训练。
- 随机打乱 优于 按顺序排列数据，全量训练 优于 增量训练。

总结

双塔模型

- 用户塔、物品塔各输出一个向量，两个向量的余弦相似度作为兴趣的预估值。
- 三种训练的方式：pointwise、pairwise、listwise。
- 正样本：用户点击过的物品。
- 负样本：整体物品（简单）、被排序淘汰的物品（困难）。

召回

- 做完训练，把物品向量存储到向量数据库，供线上最近邻查找。
- 线上召回时，给定用户 ID、用户画像，调用用户塔现算用户向量 \mathbf{a} 。
- 把 \mathbf{a} 作为 query，查询向量数据库，找到余弦相似度最高的 k 个物品向量，返回 k 个物品 ID。

更新模型

- 全量更新：今天凌晨，用昨天的数据训练整个神经网络，做 1 epoch 的随机梯度下降。
- 增量更新：用实时数据训练神经网络，只更新 ID Embedding，锁住全连接层。
- 实际的系统：
 - 全量更新 & 增量更新 相结合。
 - 每隔几十分钟，发布最新的用户 ID Embedding，供用户塔在线上计算用户向量。

双塔模型 + 自监督学习

双塔模型的问题

- 推荐系统的头部效应严重：
 - 少部分物品占据大部分点击。
 - 大部分物品的点击次数不高。
- 高点击物品的表征学得好，长尾物品的表征学得不好。
- 自监督学习：做 *data augmentation*，更好地学习长尾物品的向量

复习双塔模型

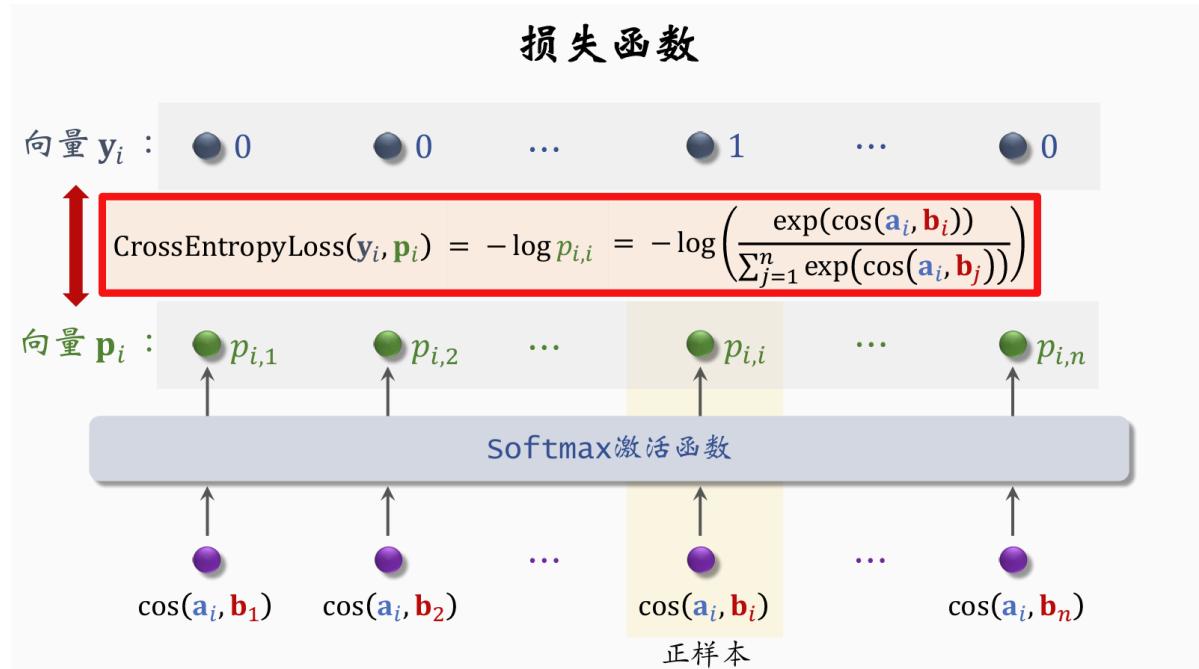
Batch内负样本

- 一个 *batch* 内有 n 对正样本。
- 组成 n 个 *list*，每个 *list* 中有 1 对正样本和 $n - 1$ 对负样本。

Listwise 训练

- 一个 *batch* 包含 n 对正样本（有点击）：
 $(\mathbf{a}_1, \mathbf{b}_1), (\mathbf{a}_2, \mathbf{b}_2), \dots, (\mathbf{a}_n, \mathbf{b}_n)$.
- 负样本： $\{(\mathbf{a}_i, \mathbf{b}_j)\}$ ，对于所有的 $i \neq j$ 。
- 鼓励 $\cos(\mathbf{a}_i, \mathbf{b}_i)$ 尽量大， $\cos(\mathbf{a}_i, \mathbf{b}_j)$ 尽量小。

损失函数



纠偏

- 物品 j 被抽样到的概率： $p_j \propto$ 点击次数
- 预估用户 i 对物品 j 的兴趣： $\cos(\mathbf{a}_i, \mathbf{b}_j)$
- 做训练的时候，把 $\cos(\mathbf{a}_i, \mathbf{b}_j)$ 替换为： $\cos(\mathbf{a}_i, \mathbf{b}_j) - \log p_j$

训练双塔模型

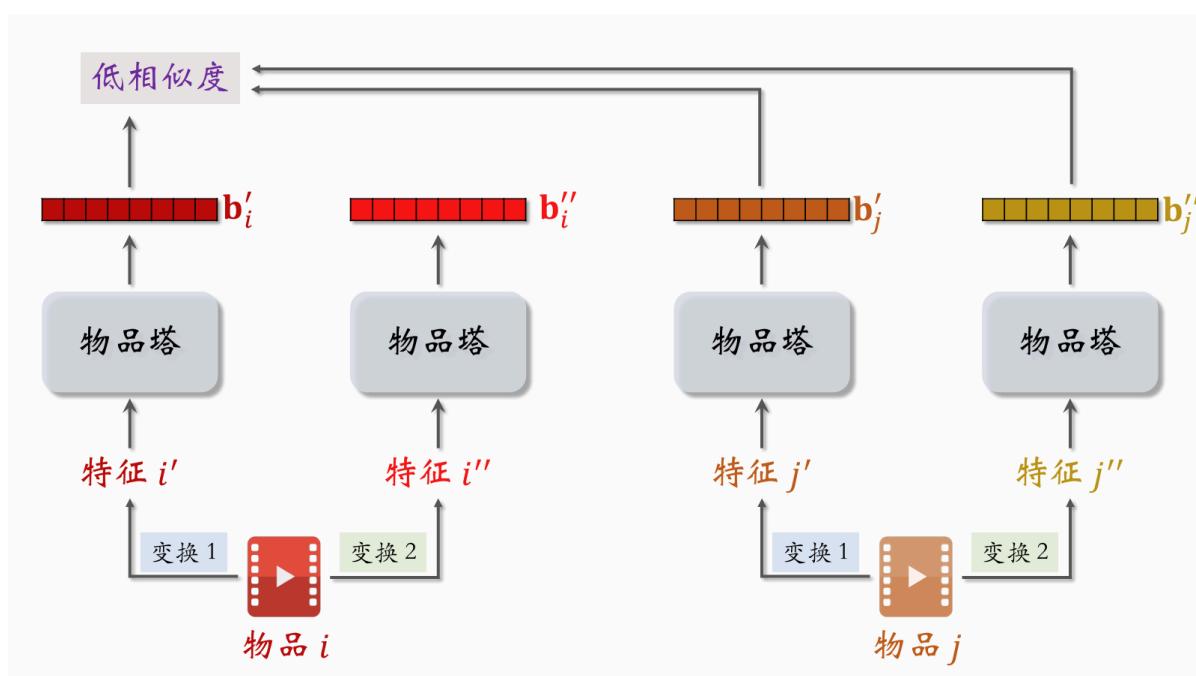
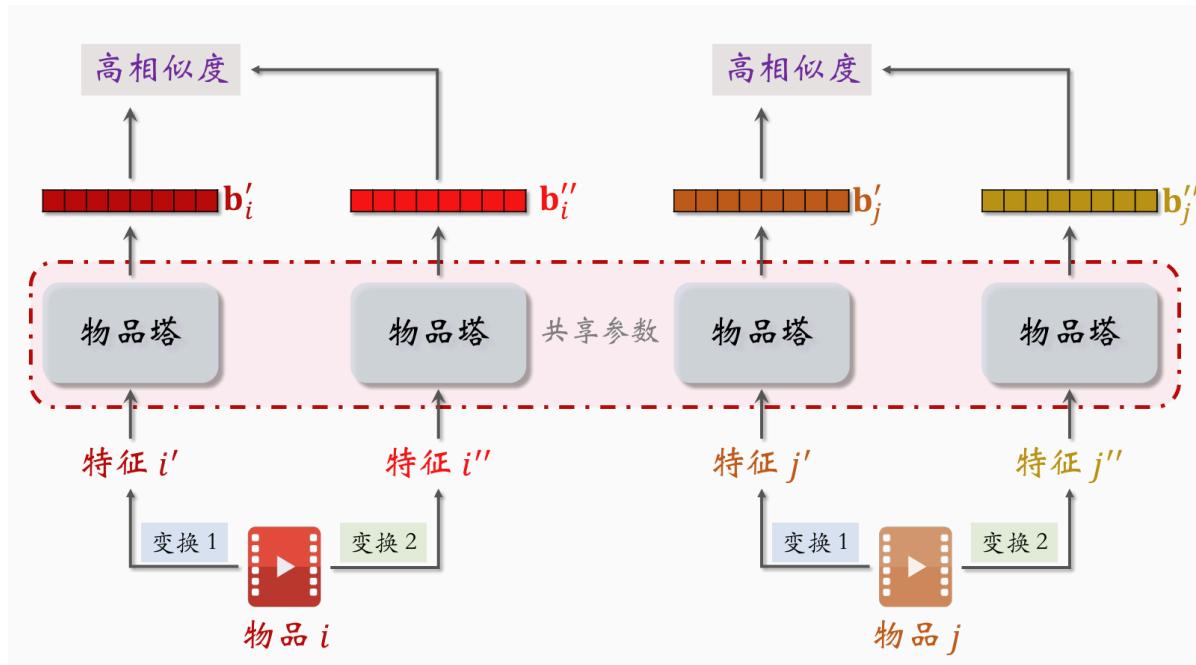
- 从点击数据中随机抽取 n 个 用户—物品二元组，组成一个 batch。
- 双塔模型的损失函数：

$$L_{\text{main}}[i] = -\log \left(\frac{\exp(\cos(\mathbf{a}_i, \mathbf{b}_i) - \log p_i)}{\sum_{j=1}^n \exp(\cos(\mathbf{a}_i, \mathbf{b}_j) - \log p_j)} \right)$$

- 做梯度下降，减少损失函数：

$$\frac{1}{n} \sum_{i=1}^n L_{\text{main}}[i]$$

自监督学习



- 物品 i 的两个向量表征 \mathbf{b}'_i 和 \mathbf{b}''_i 有较高的相似度。
- 物品 i 和 j 的向量表征 \mathbf{b}'_i 和 \mathbf{b}''_j 有较低的相似度。
- 鼓励 $\cos(\mathbf{b}'_i, \mathbf{b}''_i)$ 尽量大, $\cos(\mathbf{b}'_i, \mathbf{b}''_j)$ 尽量小。

特征变换是将一个物品的特征值的向量变换为另一个向量

比如一个物品的特征向量为（受众性别，类别，城市，职业），假设这个特征向量为（0, 5, 1.5, 9），特征变换就是把（0, 5, 1.5, 9）转换为（0.9, 7.3, 10.8, 9.6）（随便举的例子）的过程。

特征变换：Random Mask

- 随机选一些离散特征（比如 类别），把它们遮住。
- 例：
 - 某物品的 类别 特征是 $\mathcal{U} = \{\text{数码}, \text{摄影}\}$ 。
 - Mask 后的 类别 特征是 $\mathcal{U}' = \{\text{default}\}$ 。为代表缺失的默认值。
 - Mask 代表把特征中的值都丢掉。
 - 比如数码的值为 1，摄影的值为 2，缺失的默认值设置为 0。那么变换之前的类别特征值可能为 1.5，Mask 变换后的就应该为 0。
- Random mask 后不会把所有的特征都变为默认值，因为只是随机 mask 一些特征，不会 mask 所有特征。

特征变换：Dropout (仅对多值离散特征生效)

- 一个物品可以有多个 类别，那么 类别 是一个多值离散特征。
- Dropout：随机丢弃特征中 50% 的值。
- 例：
 - 某物品的 类别 特征是 $\mathcal{U} = \{\text{美妆}, \text{摄影}\}$ 。
 - Dropout 后的 类别 特征是 $\mathcal{U}' = \{\text{美妆}\}$ 。
 - 比如美妆的值为 1，摄影的值为 2。那么变换之前的类别特征值可能为 1.5，变换后的就应该为 1。

特征变换：互补特征 (complementary)

- 假设物品一共有 4 种特征：
- ID, 类别, 关键词, 城市
- 随机分成两组：{ID, 关键词} 和 {类别, 城市}
- {ID, default, 关键词, default} \Rightarrow 物品表征
- {default, 类别, default, 城市} \Rightarrow 物品表征
- 因为是同一件物品，鼓励两种物品表征向量相似

特征变换：Mask 一组关联的特征

- 一组特征相关联
 - 受众性别： $\mathcal{U} = \{\text{男}, \text{女}, \text{中性}\}$
 - 类目： $\mathcal{V} = \{\text{美妆}, \text{数码}, \text{足球}, \text{摄影}, \text{科技}, \dots\}$
 - $u = \text{女}$ 和 $v = \text{美妆}$ 同时出现的概率 $p(u, v)$ 大。
 - $u = \text{女}$ 和 $v = \text{数码}$ 同时出现的概率 $p(u, v)$ 小。
- $p(u)$ ：某特征取值为 u 的概率。
 - $p(\text{男性}) = 20\%$
 - $p(\text{女性}) = 30\%$
 - $p(\text{中性}) = 50\%$

- $p(u, v)$: 某特征取值为 u , 另一个特征取值为 v , 同时发生的概率。

- $p(\text{女性}, \text{美妆}) = 3\%$

- $p(\text{女性}, \text{数码}) = 0.1\%$

- 离线计算特征两两之间的关联, 用互信息 (*mutual information*) 衡量:
(如计算类别特征与受众性别特征的MI)

$$MI(\mathcal{U}, \mathcal{V}) = \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{V}} p(u, v) \cdot \log \frac{p(u, v)}{p(u) \cdot p(v)}$$

- 设一个物品一共有 k 种特征。离线计算特征两两之间 MI, 得到 $k \times k$ 的矩阵。
- 随机选一个特征作为种子, 找到种子最相关的 $k/2$ 种特征。
- Mask 种子及其相关的 $k/2$ 种特征, 保留其余的 $k/2$ 种特征。
- 比如某物品有四种特征{ID, 类别, 关键词, 城市}, 种子特征为关键词, 与其最相关的特征为城市, 那么mask后的特征就为{ID, 类别, default, default}
- 好处与坏处
 - 好处: 比 *random mask*、*dropout*、*互补特征*等方法效果更好。
 - 坏处: 方法复杂, 实现的难度大, 不容易维护。

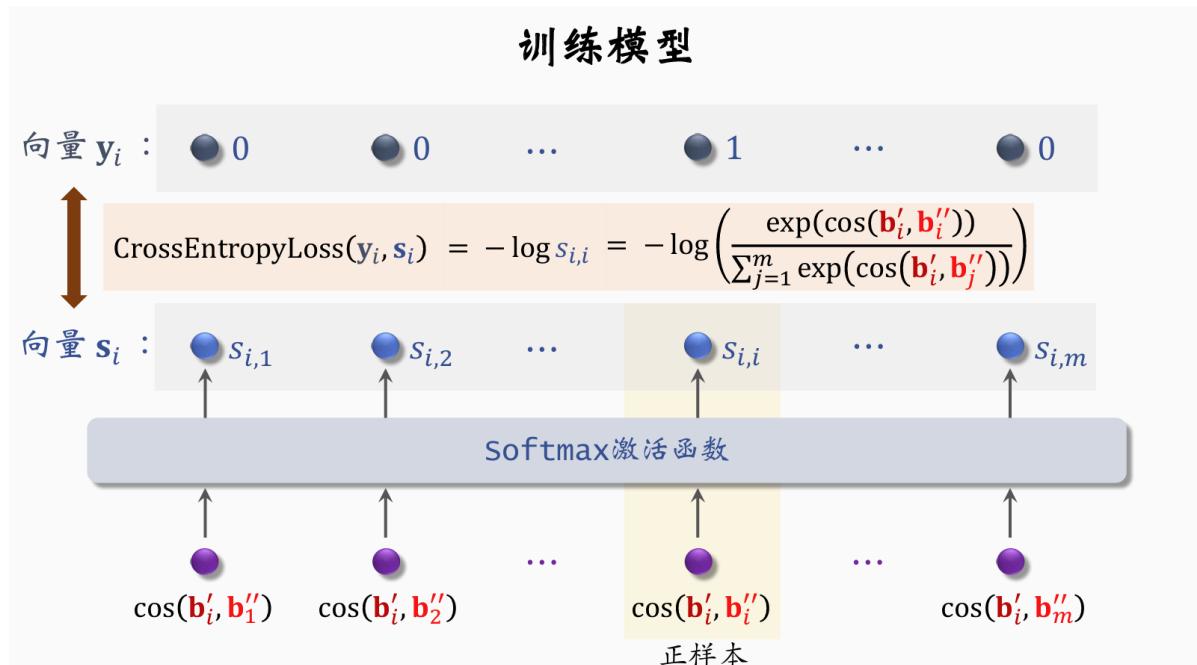
训练模型

- 从全体物品中均匀抽样, 得到 m 个物品, 作为一个 *batch*。
- 做两类特征变换, 物品塔输出两组向量:

$$\mathbf{b}'_1, \mathbf{b}'_2, \dots, \mathbf{b}'_m \quad \text{和} \quad \mathbf{b}''_1, \mathbf{b}''_2, \dots, \mathbf{b}''_m$$

- 第 i 个物品的损失函数:

$$L_{\text{self}}[i] = -\log \left(\frac{\exp(\cos(\mathbf{b}'_i, \mathbf{b}''_i))}{\sum_{j=1}^m \exp(\cos(\mathbf{b}'_i, \mathbf{b}''_j))} \right)$$



- 自监督学习的损失函数:

$$L_{\text{self}}[i] = -\log \left(\frac{\exp(\cos(\mathbf{b}'_i, \mathbf{b}''_i))}{\sum_{j=1}^m \exp(\cos(\mathbf{b}'_i, \mathbf{b}''_j))} \right)$$

- 做梯度下降，减少自监督学习的损失：

$$\frac{1}{m} \sum_{i=1}^m L_{\text{self}}[i]$$

总结

- 双塔模型学不好低曝光物品的向量表征。
- 自监督学习：
 - 对物品做随机特征变换。
 - 特征向量 \mathbf{b}'_i 和 \mathbf{b}''_i 相似度高（相同物品）。
 - 特征向量 \mathbf{b}'_i 和 \mathbf{b}''_j 相似度低（不同物品）。
- 实验效果：低曝光物品、新物品的推荐变得更准。
- 对点击做随机抽样，得到 n 对 用户—物品二元组，作为一个 batch。
- 从全体 物品 中均匀抽样，得到 m 个 物品，作为一个 batch。
- 做梯度下降，使得损失减少：

$$\frac{1}{n} \sum_{i=1}^n L_{\text{main}}[i] + \alpha \cdot \frac{1}{m} \sum_{j=1}^m L_{\text{self}}[j].$$

Deep Retrieval 召回

- 经典的双塔模型把用户、物品表示为向量，线上做最近邻查找。
- Deep Retrieval 把物品表征为路径 (path)，线上查找用户最匹配的路径。
- Deep Retrieval 类似于阿里的 TDM。

Outline

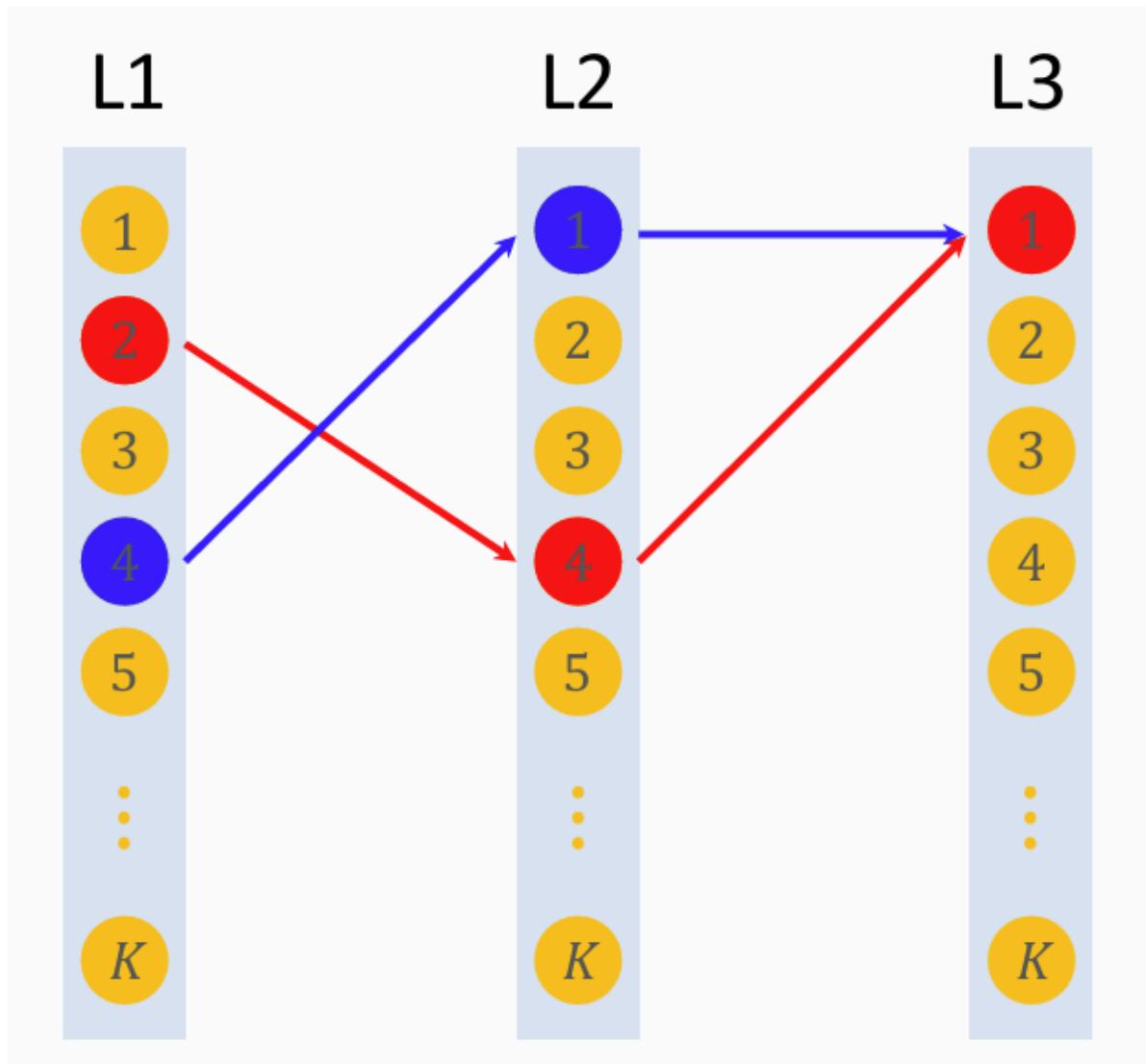
1. 索引：
 - 路径 \rightarrow List<物品>
 - 物品 \rightarrow List<路径>
2. 预估模型：神经网络预估用户对路径的兴趣。
3. 线上召回：用户 \rightarrow 路径 \rightarrow 物品。
4. 训练：
 - 学习神经网络参数。
 - 学习物品表征（物品 \rightarrow 路径）

索引

物品表征为路径

- 深度： $depth = 3$ 。也就是层数。
- 宽度： $width = K$ 。
- 一个物品表示为一条路径 (path)，比如 [2, 4, 1]。

- 一个物品可以表示为多条路径，比如 $\{[2, 4, 1], [4, 1, 1]\}$ 。



物品到路径的索引

索引: item \rightarrow List(path)

- 一个物品对应多条路径。
- 用 3 个节点表示一条路径: path = $[a, b, c]$ 。

索引: path \rightarrow List(item)

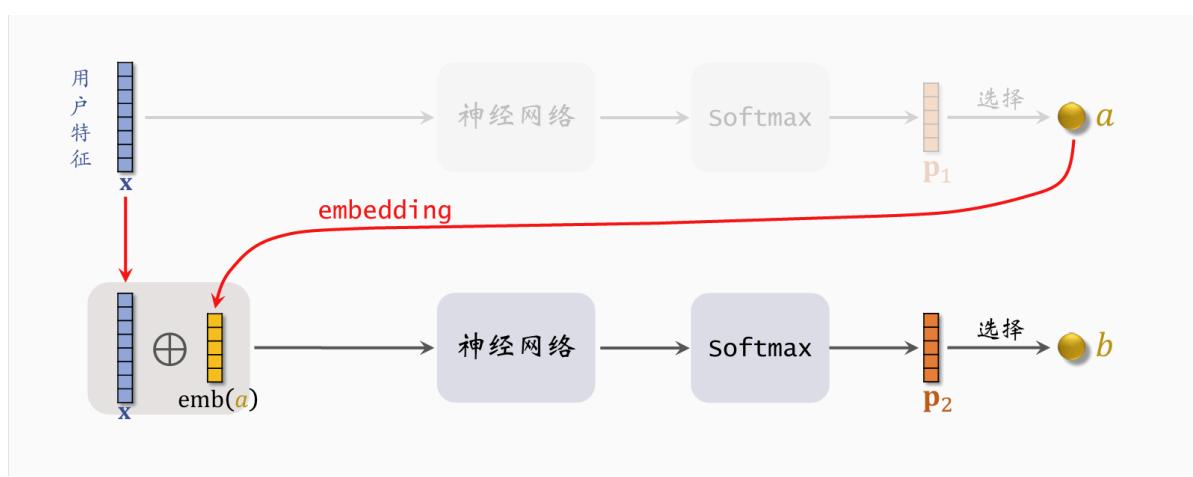
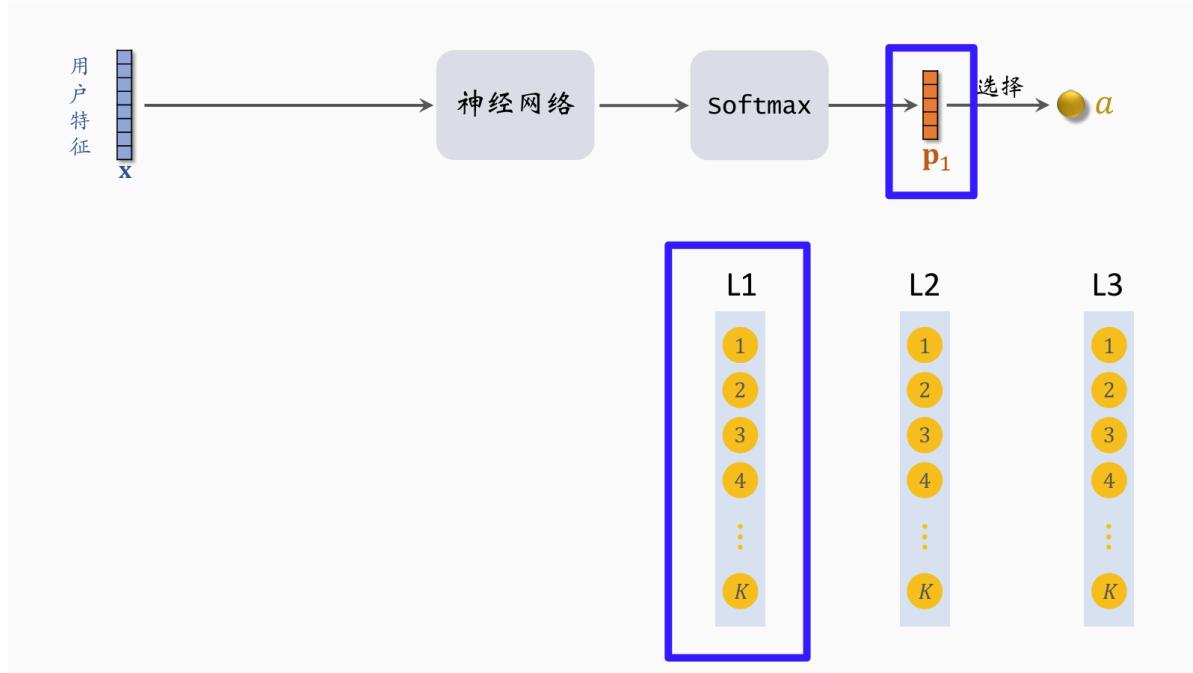
- 一条路径对应多个物品。

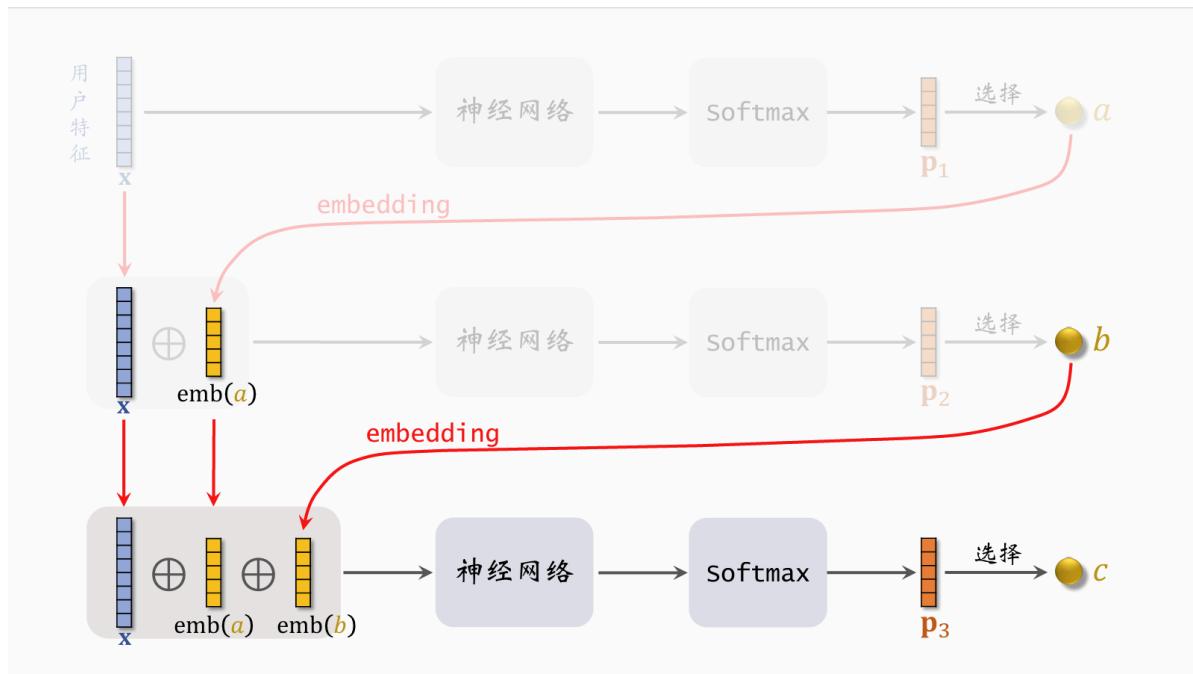
预估模型

预估用户对路径的兴趣

- 用 3 个节点表示一条路径: path = $[a, b, c]$ 。
- 给定用户特征 \mathbf{x} , 预估用户对节点 a 的兴趣 $p_1(a|\mathbf{x})$ 。
- 给定 \mathbf{x} 和 a , 预估用户对节点 b 的兴趣 $p_2(b|a; \mathbf{x})$ 。
- 给定 \mathbf{x}, a, b , 预估用户对节点 c 的兴趣 $p_3(c|a, b; \mathbf{x})$ 。
- 预估用户对 path = $[a, b, c]$ 兴趣:

$$p(a, b, c|\mathbf{x}) = p_1(a|\mathbf{x}) \times p_2(b|a; \mathbf{x}) \times p_3(c|a, b; \mathbf{x})$$





以此类推得到节点 c。

线上召回

召回：用户 \rightarrow 路径 \rightarrow 物品

- 第一步：给定用户特征，用 *beam search* 召回一批路径。
- 第二步：利用索引 “path \rightarrow List(item)”，召回一批物品。
- 第三步：对物品做打分和排序，选出一个子集。

Beam Search

- 假设有 3 层，每层 K 个节点，那么一共有 K^3 条路径。
- 用神经网络给所有 K^3 条路径打分，计算量太大。
- 用 beam search，可以减小计算量。
- 需要设置超参数 beam size。

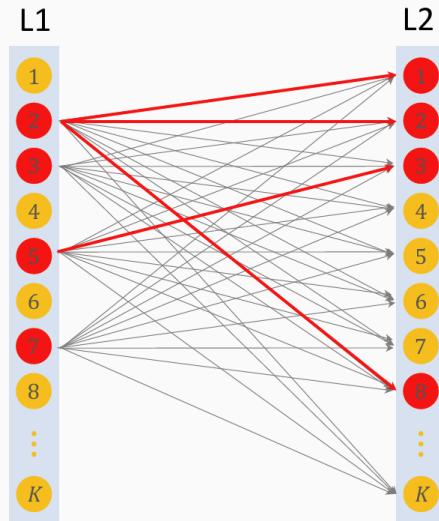
Beam Search (size = 4)

L1

1	$p_1(1 \mathbf{x})$
2	$p_1(2 \mathbf{x})$
3	$p_1(3 \mathbf{x})$
4	$p_1(4 \mathbf{x})$
5	$p_1(5 \mathbf{x})$
6	$p_1(6 \mathbf{x})$
7	$p_1(7 \mathbf{x})$
8	$p_1(8 \mathbf{x})$
:	:
K	$p_1(K \mathbf{x})$

根据神经网络对 L1 层的 k 个节点打的分选出四个分数最高的节点。

Beam Search (size = 4)



- 对于每个被选中的节点 a ，计算用户对路径 $[a, b]$ 的兴趣：

$$p(a, b|x) = p_1(a|x) \times p_2(b|a; x).$$

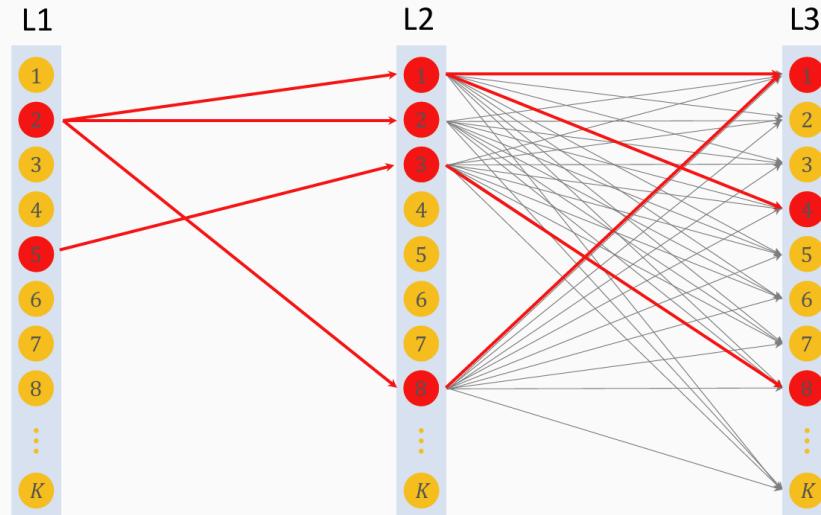
- 算出 $4 \times K$ 个分数，每个分数对应一条路径，选出分数 top 4 路径。

对于每个被选中的节点 a ，计算用户对路径 $[a, b]$ 的兴趣：

$$p(a, b|x) = p_1(a|x) \times p_2(b|a; x)$$

算出 $4 \times K$ 个分数，每个分数对应一条路径，选出分数 top 4 路径。

Beam Search (size = 4)



对于每个被选中的节点 a, b ，计算用户对路径 $[a, b, c]$ 的兴趣：

$$p(a, b, c|x) = p(a, b|x) \times p_3(c|a, b; x)$$

再算出 $4 \times K$ 个分数，每个分数对应一条路径，选出分数 top 4 路径。

Beam Search

- 用户对 $\text{path} = [a, b, c]$ 兴趣：

$$p(a, b, c|x) = p_1(a|x) \times p_2(b|a; x) \times p_3(c|a, b; x)$$

- 最优的路径：

$$[a^*, b^*, c^*] = \arg \max_{a, b, c} p(a, b, c|x)$$

- 贪心算法 (*beam size = 1*) 选中的路径 $[a, b, c]$ 未必是最优的路径。

线上召回

- **第一步**: 给定用户特征, 用神经网络做预估, 用 *beam search* 召回一批路径。
- **第二步**: 利用索引, 召回一批物品。
 - 查看索引 $\text{path} \rightarrow \text{List}(\text{item})$ 。
 - 每条路径对应多个物品。
- **第三步**: 对物品做排序, 选出一个子集。

训练

同时学习神经网络参数和物品表征

- 神经网络 $p(a, b, c | \mathbf{x})$ 预估用户对路径 $[a, b, c]$ 的兴趣。
- 把一个物品表征为多条路径 $\{[a, b, c]\}$, 建立索引:
 - $\text{item} \rightarrow \text{List}(\text{path})$,
 - $\text{path} \rightarrow \text{List}(\text{item})$ 。
- 正样本 $(\text{user}, \text{item})$: $\text{click}(\text{user}, \text{item}) = 1$ 。

学习神经网络参数

- 物品表征为 J 条路径: $[a_1, b_1, c_1], \dots, [a_J, b_J, c_J]$ 。
- 用户对路径 $[a, b, c]$ 的兴趣:

$$p(a, b, c | \mathbf{x}) = p_1(a | \mathbf{x}) \times p_2(b | a; \mathbf{x}) \times p_3(c | a, b; \mathbf{x})$$

- 如果用户点击过物品, 说明用户对 J 条路径全都感兴趣。
- 应该让 $\sum_{j=1}^J p(a_j, b_j, c_j | \mathbf{x})$ 变大。
- 损失函数:
对 J 条路径的兴趣的加和越大, 损失函数就越小。

$$\text{loss} = -\log \left(\sum_{j=1}^J p(a_j, b_j, c_j | \mathbf{x}) \right)$$

学习物品表征

- 用户 user 对路径 $\text{path} = [a, b, c]$ 的兴趣记作:

$$p(\text{path} | \text{user}) = p(a, b, c | \mathbf{x})$$

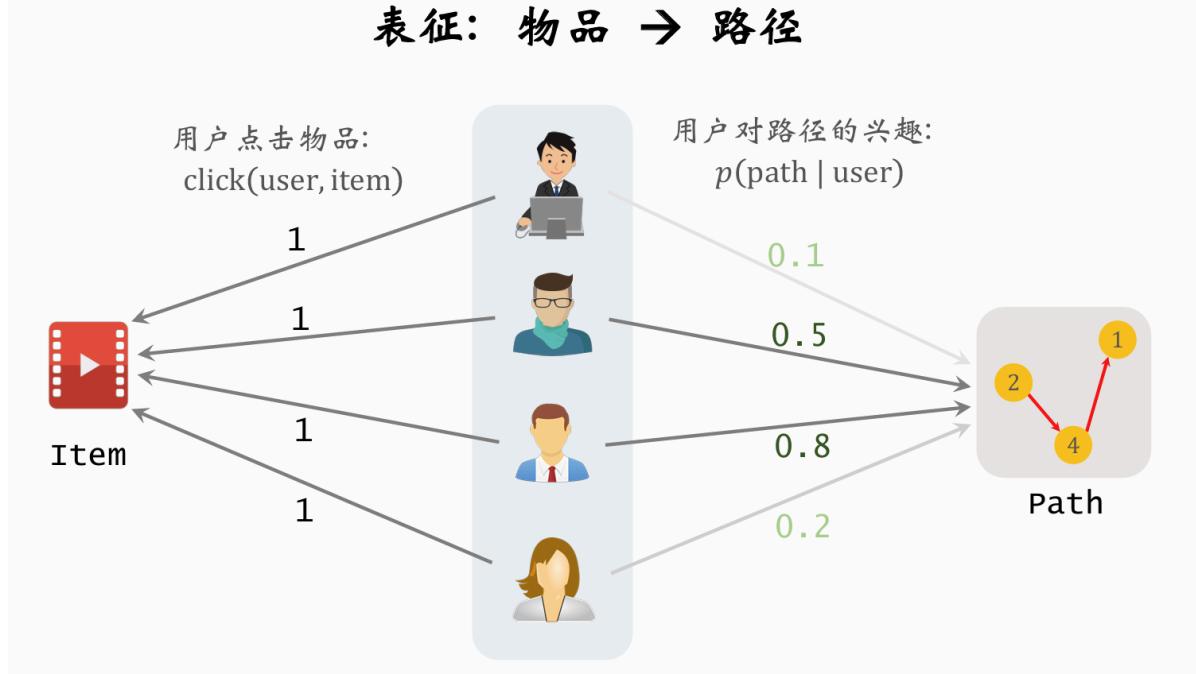
- 物品 item 与路径 path 的相关性:

$$\text{score}(\text{item}, \text{path}) = \sum_{\text{user}} p(\text{path} | \text{user}) \times \text{click}(\text{user}, \text{item})$$

$\text{click}(\text{user}, \text{item})$ 代表用户是否点击过物品, 点击过就是 1, 没点击过就是 0。

- 根据 $\text{score}(\text{item}, \text{path})$ 选出 J 条路径作为 item 的表征。

表征：物品 → 路径



- 选出 J 条路径 $\Pi = \{\text{path}_1, \dots, \text{path}_J\}$, 作为物品的表征。

- 损失函数** (选择与 $item$ 高度相关的 $path$) :

$$\text{loss}(\text{item}, \Pi) = -\log \left(\sum_{j=1}^J \text{score}(\text{item}, \text{path}_j) \right)$$

- 正则项** (避免过多的 $item$ 集中在一条 $path$ 上) :

$$\text{reg}(\text{path}_j) = (\text{number of items on path}_j)^4.$$

用贪心算法更新路径

- 假设已经把物品表征为 J 条路径 $\Pi = \{\text{path}_1, \dots, \text{path}_J\}$ 。现在要更新 Π 中的路径
- 每次固定 $\{\text{path}_i\}_{i \neq l}$, 并从未被选中的路径中, 选出一条作为新的 path_l :
未被选择的路径不限于 J 条路径中, 而是从外界选取路径。选择的范围可以是先前计算的 $\text{score}(\text{item}, \text{path})$ 较高的 N 条路径。 $\text{loss}(\text{item}, \Pi)$ 代表 $\{\text{path}_i\}$ 与 path_l 构成的路径集合与物品的损失函数。 $\text{reg}(\text{path}_l)$ 是为了防止一条路径上的物品数量太多。

$$\text{path}_l \leftarrow \arg \min_{\text{path}_l} \text{loss}(\text{item}, \Pi) + \alpha \cdot \text{reg}(\text{path}_l)$$

- 选中的路径有较高的分数 $\text{score}(\text{item}, \text{path}_l)$, 而且路径上的物品数量不会太多。

对比

更新神经网络

- 神经网络判断用户对路径的兴趣:

$$p(\text{path} | \mathbf{x})$$

- 训练所需的数据:

1. “物品 → 路径”的索引,
2. 用户点击过的物品。

- 如果用户点击过物品, 且物品对应路径 path , 则更新神经网络参数使 $p(\text{path} | \mathbf{x})$ 变大。

更新物品的表征

- 判断物品与路径的相关性：

物品 \leftarrow 用户点击过物品 用户 \longrightarrow 神经网络的打分 路径

- 让每个物品关联 J 条路径：

- 物品和路径要有很高的相关性。
- 一条路径上不能有过多的物品。

总结

召回：用户 \rightarrow 路径 \rightarrow 物品

- 给定用户特征 \mathbf{x} ，用神经网络预估用户对路径 $\text{path} = [a, b, c]$ 的兴趣，分数记作 $p(\text{path} \mid \mathbf{x})$ 。
- 用 beam search 寻找分数 $p(\text{path} \mid \mathbf{x})$ 最高的 s 条 path。
- 利用索引 “ $\text{path} \rightarrow \text{List}\{\text{item}\}$ ” 召回每条路径上的 n 个物品。
- 一共召回 $s \times n$ 个物品，对物品做初步排序，返回分数最高的若干物品。

训练：同时学习 用户—路径 和 物品—路径 的关系

- 一个物品被表征为 J 条路径： $\text{path}_1, \dots, \text{path}_J$ 。
- 如果用户点击过物品，则更新神经网络参数，使分数增大：

$$\sum_{j=1}^J p(\text{path}_j \mid \mathbf{x}).$$

- 如果用户对路径的兴趣分数 $p(\text{path} \mid \mathbf{x})$ 较高，且用户点击过物品 item，则 item 与 path 具有相关性。
- 寻找与 item 最相关的 J 条 path，且避免一条路径上物品过多。

其他召回项目

地理位置召回

GeoHash召回

- 用户可能对附近发生的事感兴趣。
- GeoHash：对经纬度的编码，地图上一个长方形区域。
- 索引：GeoHash \rightarrow 优质笔记列表（按时间倒排）。
- 这条召回通道没有个性化。

同城召回

- 用户可能对同城发生的事感兴趣。
- 索引：城市 \rightarrow 优质笔记列表（按时间倒排）。
- 这条召回通道没有个性化。

作者召回

关注作者召回

- 用户对关注的作者发布的笔记感兴趣。
- 索引：

用户 → 关注的作者

作者 → 发布的笔记

- 召回：

用户 → 关注的作者 → 最新的笔记

有交互的作者召回

- 如果用户对某笔记感兴趣（点赞、收藏、转发），那么用户可能对该作者的其他笔记感兴趣。
- 索引：
用户 → 有交互的作者
- 召回：
用户 → 有交互的作者 → 最新的笔记

相似作者召回

- 如果用户喜欢某作者，那么用户喜欢相似的作者。
- 索引：
作者 → 相似作者 (k 个作者)
- 召回：
用户 → 感兴趣的作者 (n 个作者) → 相似作者 (nk 个作者) → 最新的笔记 (nk 篇笔记)

缓存召回

缓存召回

想法：复用前 n 次推荐精排的结果。

- 背景：
 - 精排输出几百篇笔记，送入重排。
 - 重排做多样性抽样，选出几十篇。
 - 精排结果一大半没有曝光，被浪费。
- 精排前 50，但是没有曝光的，缓存起来，作为一条召回通道。

缓存大小固定，需要退场机制。

- 一旦笔记成功曝光，就从缓存退场。
- 如果超出缓存大小，就移除最先进入缓存的笔记。
- 笔记最多被召回 10 次，达到 10 次就退场。
- 每篇笔记最多保存 3 天，达到 3 天就退场。

曝光过滤 & Bloom Filter (布隆过滤器)

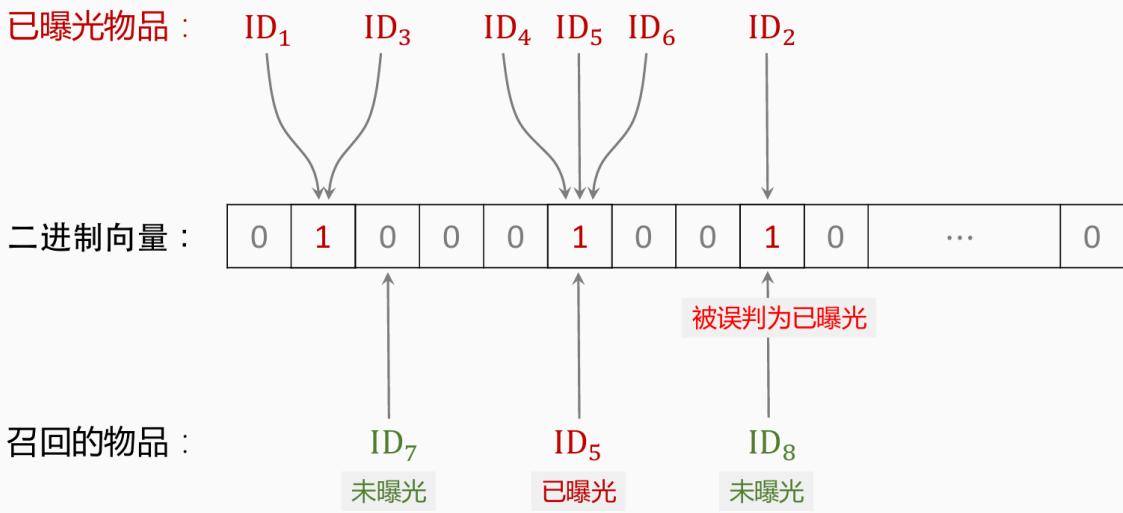
曝光过滤问题

- 如果用户看过某个物品，则不再把该物品曝光给该用户。
- 对于每个用户，记录已经曝光给他的物品。（小红书只召回 1 个月以内的笔记，因此只需要记录每个用户最近 1 个月的曝光历史。）
- 对于每个召回的物品，判断它是否已经给该用户曝光过，排除掉曾经曝光过的物品。
- 一位用户看过 n 个物品，本次召回 r 个物品，如果暴力对比，需要 $O(nr)$ 的时间。

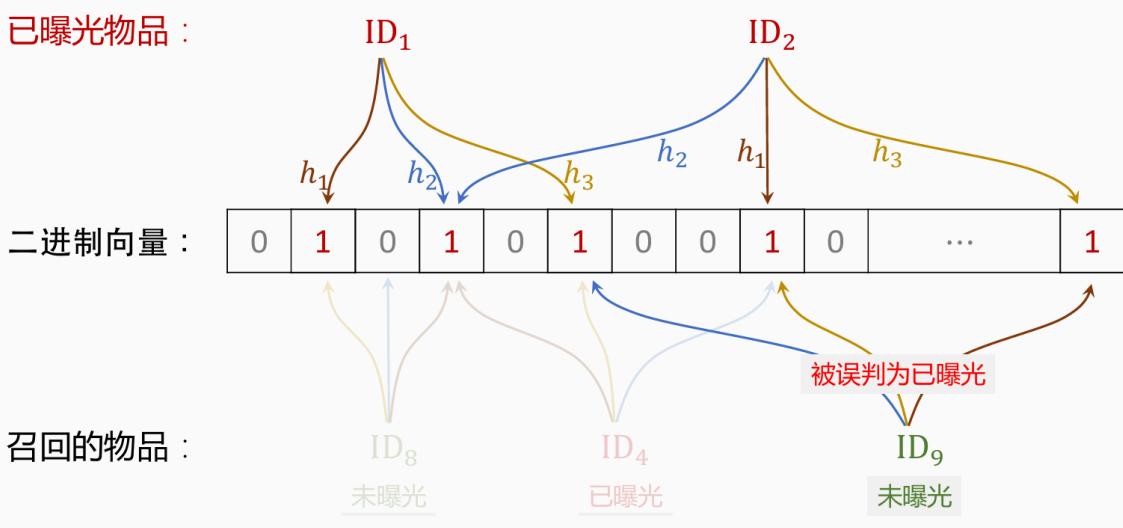
Bloom Filter

- Bloom filter 判断一个物品 ID 是否在已曝光的物品集合中。
- 如果判断为 no, 那么该物品一定不在集合中。
- 如果判断为 yes, 那么该物品很可能在集合中。 (可能误伤, 错误判断未曝光物品为已曝光, 将其过滤掉。)
- Bloom filter 把物品集合表征为一个 m 维二进制向量。
- 每个用户有一个曝光物品的集合, 表征为一个向量, 需要 m bit 的存储。
- Bloom filter 有 k 个哈希函数, 每个哈希函数把物品 ID 映射成介于 0 和 $m - 1$ 之间的整数。

Bloom Filter ($k = 1$)



Bloom Filter ($k = 3$)

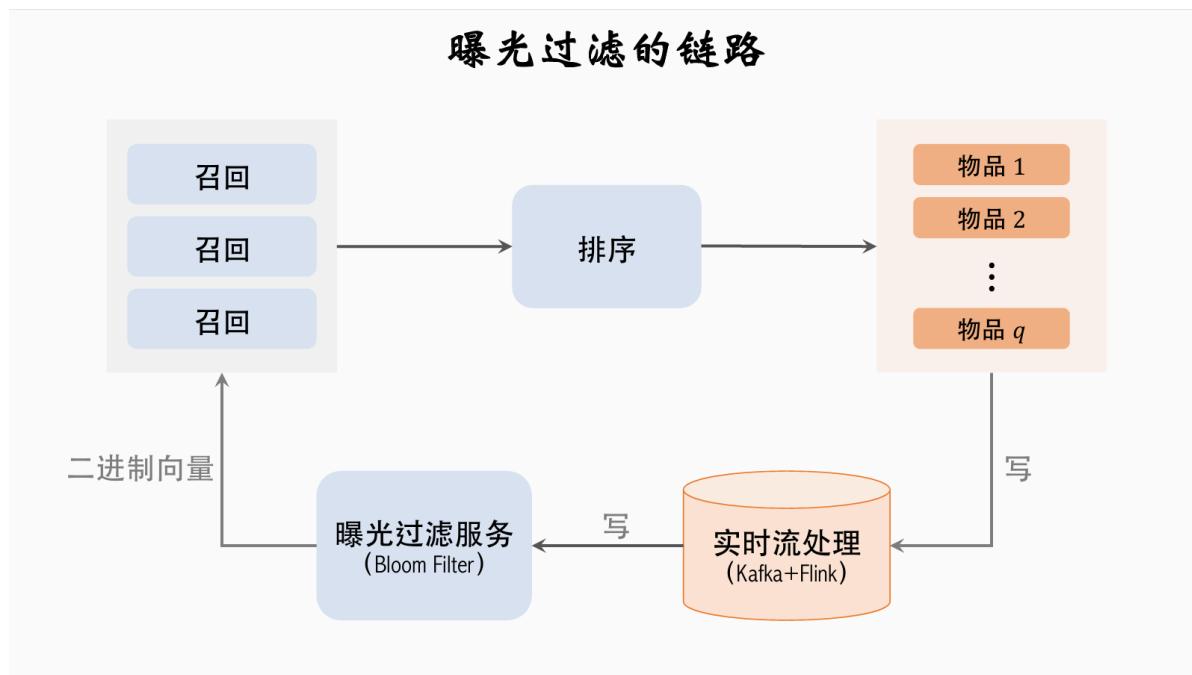


Bloom Filter

- 曝光物品集合大小为 n , 二进制向量维度为 m , 使用 k 个哈希函数。
- Bloom filter 误伤的概率为 $\delta \approx \left(1 - \exp\left(-\frac{kn}{m}\right)\right)^k$ 。
 - n 越大, 向量中的 1 越多, 误伤概率越大。 (未曝光物品的 k 个位置恰好都是 1 的概率大。)

- m 越大，向量越长，越不容易发生哈希碰撞。
- k 太大、太小都不好， k 有最优取值。
- 设定可容忍的误伤概率为 δ ，那么最优参数为：

$$k = 1.44 \cdot \ln\left(\frac{1}{\delta}\right), \quad m = 2n \cdot \ln\left(\frac{1}{\delta}\right)$$



Bloom Filter的缺点

- Bloom filter 把物品的集合表示成一个二进制向量。
- 每往集合中添加一个物品，只需要把向量 k 个位置的元素置为 1。（如果原本就是 1，则不变。）
- Bloom filter 只支持添加物品，不支持删除物品。从集合中移除物品，无法消除它对向量的影响。
- 每天都需要从物品集合中移除年龄大于 1 个月的物品。（超龄物品不可能被召回，没必要把它们记录在 Bloom filter，降低 n 可以降低误伤率。）

排序

多目标排序模型

用户一笔记的交互

- 对于每篇笔记，系统记录：
 - **曝光次数** (*number of impressions*)
 - **点击次数** (*number of clicks*)
 - **点赞次数** (*number of likes*)
 - **收藏次数** (*number of collects*)
 - **转发次数** (*number of shares*)
- **点击率** = 点击次数/曝光次数
- **点赞率** = 点赞次数/点击次数

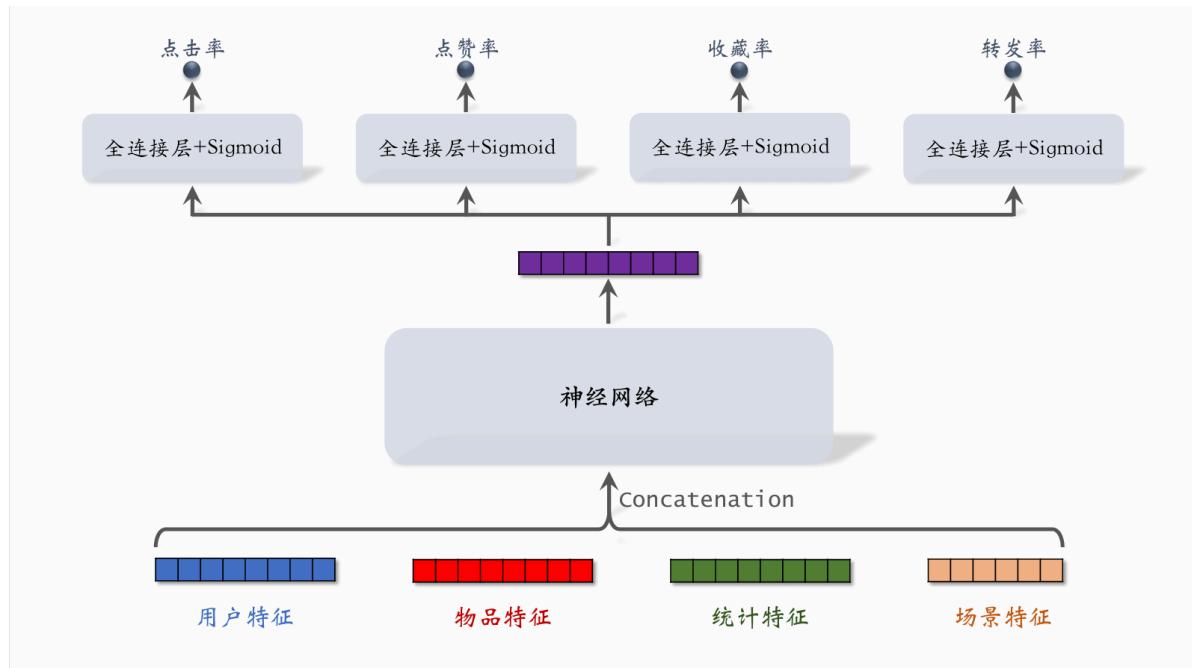
- 收藏率 = 收藏次数/点击次数

- 转发率 = 转发次数/点击次数

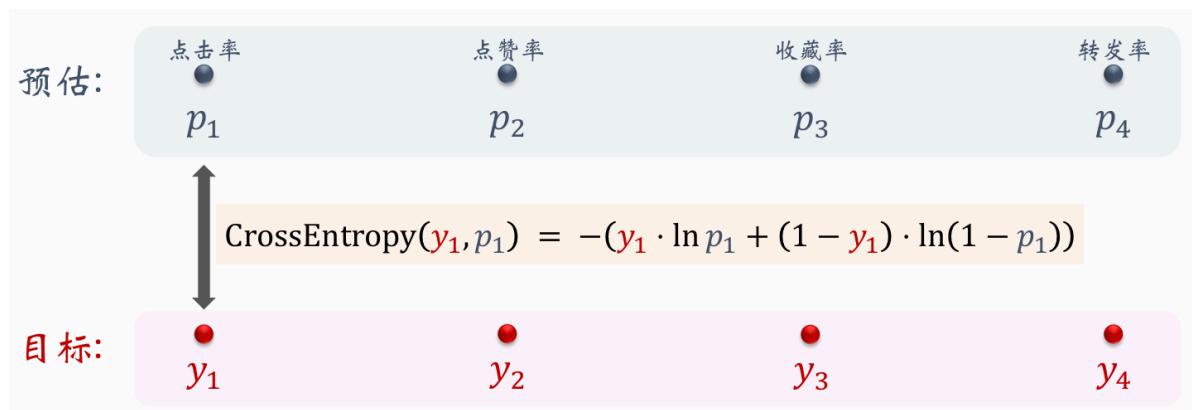
排序的依据

- 排序模型预估点击率、点赞率、收藏率、转发率等多种分数。
- 融合这些预估分数。 (比如加权和。)
- 根据融合的分数做排序、截断。

多目标模型



- 用户特征：如用户ID，用户画像
- 物品特征：如物品ID，物品画像，作者信息
- 统计特征：如用户在过去一段时间内曝光了多少篇笔记，点击了多少篇笔记，点赞了多少篇笔记。
物品在过去一段时间内获得了多少曝光机会，被点击了多少次，点赞了多少次
- 场景特征：如当前的时间，用户所在的地点



训练：

- 对于点击率来说，模型实际上就是根据点击率来判断一个物品是否为被点击物品。这是一个二元分类问题，因此用交叉熵损失函数。
- 总的损失函数： $\sum_{i=1}^4 \alpha_i \cdot \text{CrossEntropy}(y_i, p_i)$ 。
- 对损失函数求梯度，做梯度下降更新参数。

训练

- 困难：类别不平衡。
 - 每 100 次曝光，约有 10 次点击，90 次无点击。
 - 每 100 次点击，约有 10 次收藏，90 次无收藏。
 - 负样本与正样本数量差距悬殊，多出的负样本意义不大，浪费计算资源。
- 解决方案：负样本降采样 (*down-sampling*)
 - 保留一小部分负样本。
 - 让正负样本数量平衡，节约计算。

预估值校准

- 正样本、负样本数量为 n_+ 和 n_- 。
- 对负样本做降采样，抛弃一部分负样本。
- 使用 $\alpha \cdot n_-$ 个负样本， $\alpha \in (0, 1)$ 是采样率。
- 由于负样本变少，**预估点击率** 大于 **真实点击率**。而且 α 越小，负样本越少，模型对点击率高估就越严重。

- **真实点击率**：

$$p_{\text{true}} = \frac{n_+}{n_+ + n_-} \quad (\text{期望})$$

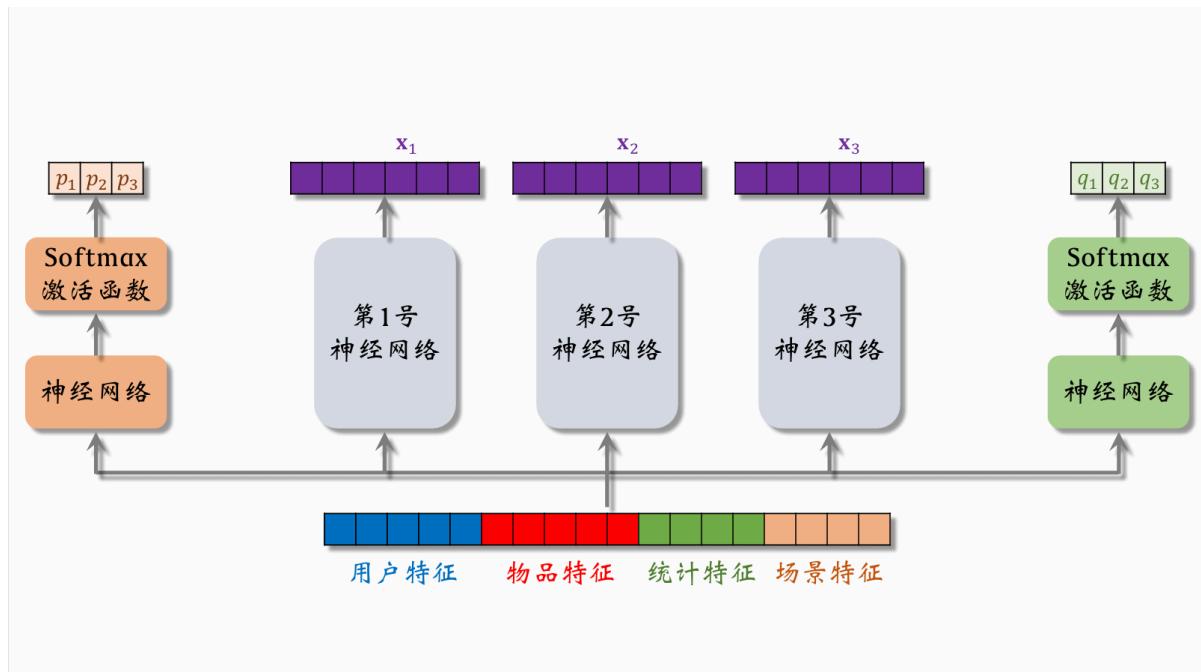
- **预估点击率**：

$$p_{\text{pred}} = \frac{n_+}{n_+ + \alpha \cdot n_-} \quad (\text{期望})$$

- 由上面两个等式可得校准公式：

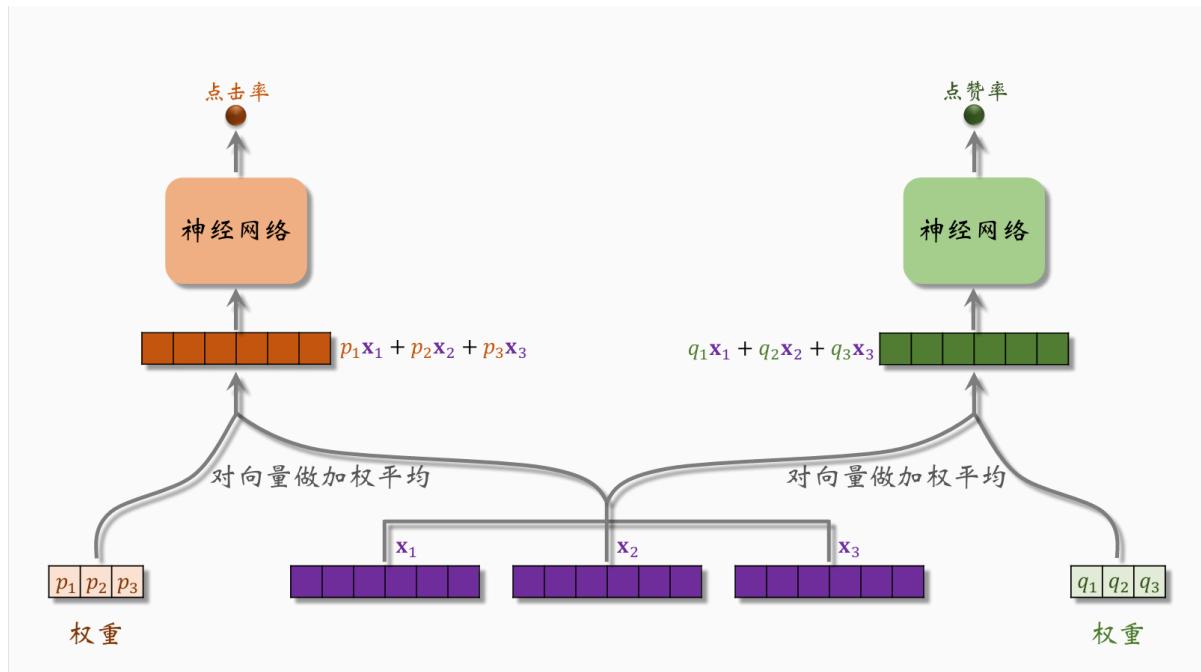
$$p_{\text{true}} = \frac{\alpha \cdot p_{\text{pred}}}{(1 - p_{\text{pred}}) + \alpha \cdot p_{\text{pred}}}.$$

Multi-gate Mixture-of-Experts (MMoE)



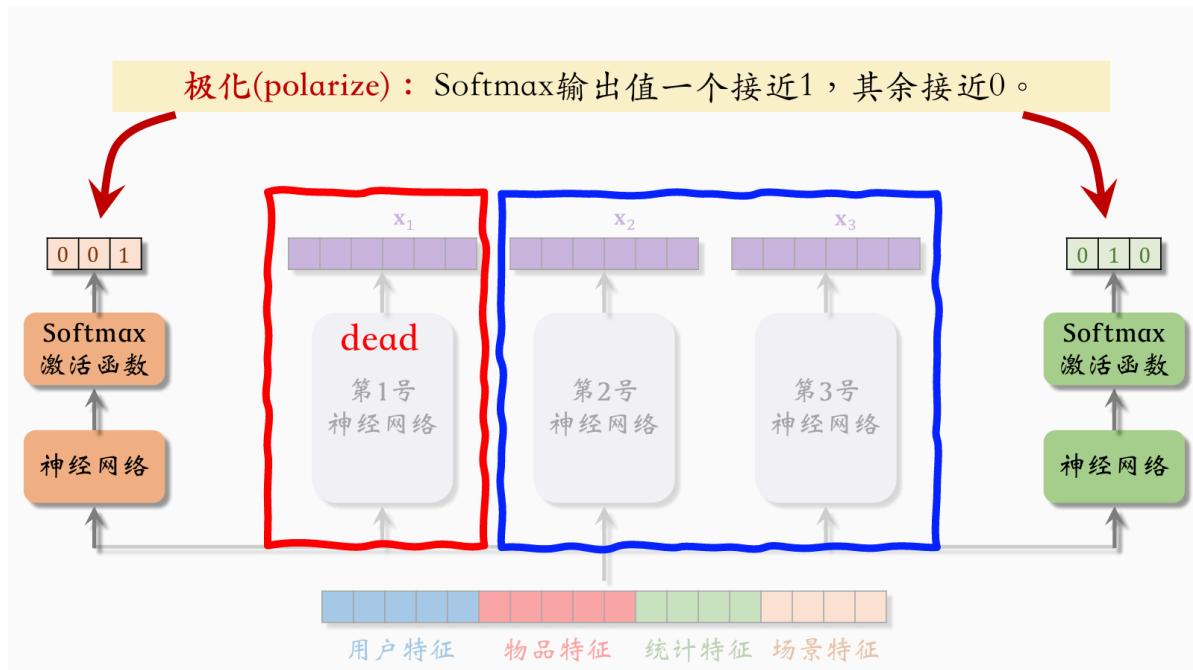
1号, 2号, 3号神经网络分别是专家神经网络。

将用户特征, 物品特征, 统计特征和行为特征输入到左侧的神经网络中, 再经过 softmax 激活函数得到一个权重向量, 这种神经网络也叫做门控神经网络。向量中的三个值 p_1, p_2, p_3 分别代表着对 x_1, x_2, x_3 的权重值。右侧神经网络同理。



将左侧门控神经网络输出的权重向量与专家神经网络生成的特征向量结合, 输入到左侧的任务神经网络中, 预测点击率。右侧同理。

极化现象 (Polarization)



当左侧门控神经网络输出向量中 3 号专家神经网络的权重接近 1，其余接近 0。右侧门控神经网络输出向量中 2 号专家神经网络的权重接近 1，其余接近 0。这样加权过后就会导致 1 号神经网络的输出向量没有参与模型工作。应该尽量避免这种情况。

解决极化问题

- 如果有 n 个“专家”，那么每个 softmax 的输入和输出都是 n 维向量。
- 在训练时，对 softmax 的输出使用 dropout。**
 - Softmax 输出的 n 个数值被 mask 的概率都是 10%。
 - 每个“专家”被随机丢弃的概率都是 10%。

预估分数的融合

简单的加权和

$$p_{\text{click}} + w_1 \cdot p_{\text{like}} + w_2 \cdot p_{\text{collect}} + \dots$$

点击率乘以其他项的加权和

$$p_{\text{click}} \cdot (1 + w_1 \cdot p_{\text{like}} + w_2 \cdot p_{\text{collect}} + \dots)$$

- $p_{\text{click}} = \frac{\text{点击}}{\text{曝光}}$
- $p_{\text{like}} = \frac{\text{点赞}}{\text{点击}}$

国内某短视频 APP 的融合分公式

- 根据预估时长 p_{time} ，对 n 篇候选视频做排序。
- 如果某视频排名第 r_{time} ，则它得分 $\frac{1}{r_{\text{time}}^{\alpha} + \beta}$ 。
- 对点击、点赞、转发、评论等预估分数做类似处理。
- 最终融合分数：($\alpha_{1,2,3,\dots}$ 为超参数)

$$\frac{w_1}{r_{\text{time}}^{\alpha_1} + \beta_1} + \frac{w_2}{r_{\text{click}}^{\alpha_2} + \beta_2} + \frac{w_3}{r_{\text{like}}^{\alpha_3} + \beta_3} + \dots$$

某电商的融合分公式

- 电商的转化流程：
曝光 → 点击 → 加购物车 → 付款
- 模型预估： p_{click} 、 p_{cart} 、 p_{pay} 。
- 最终融合分数：（ $\alpha_{1,2,3,4,\dots}$ 为超参数）

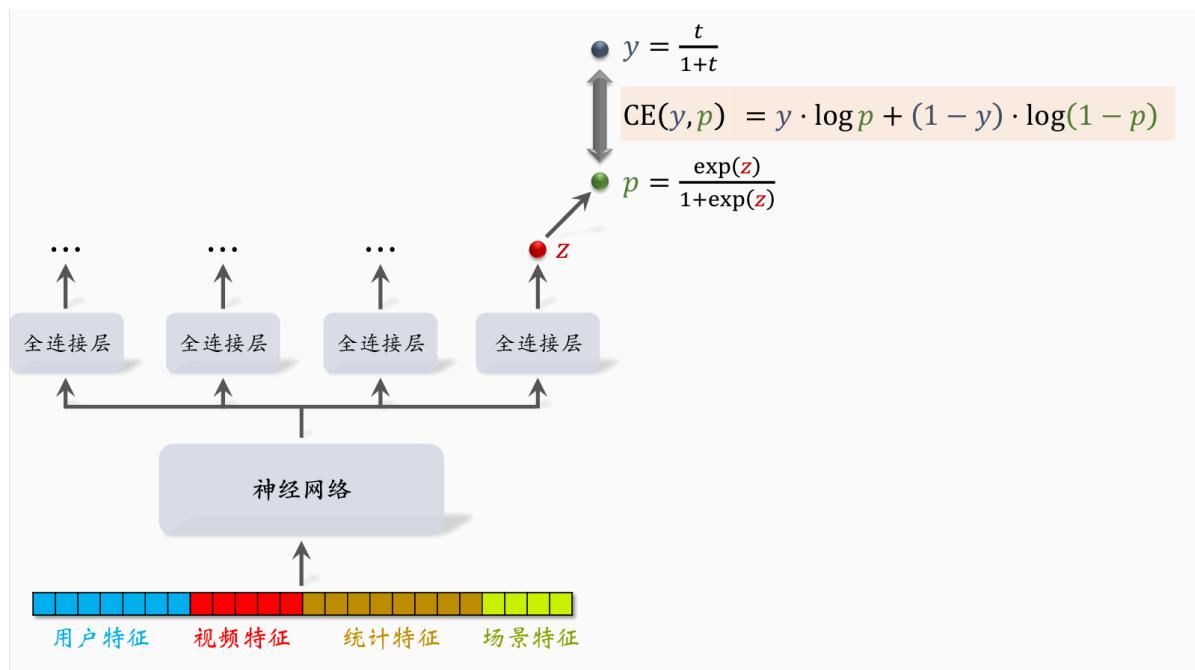
$$p_{\text{click}}^{\alpha_1} \times p_{\text{cart}}^{\alpha_2} \times p_{\text{pay}}^{\alpha_3} \times \text{price}^{\alpha_4}$$

视频播放建模

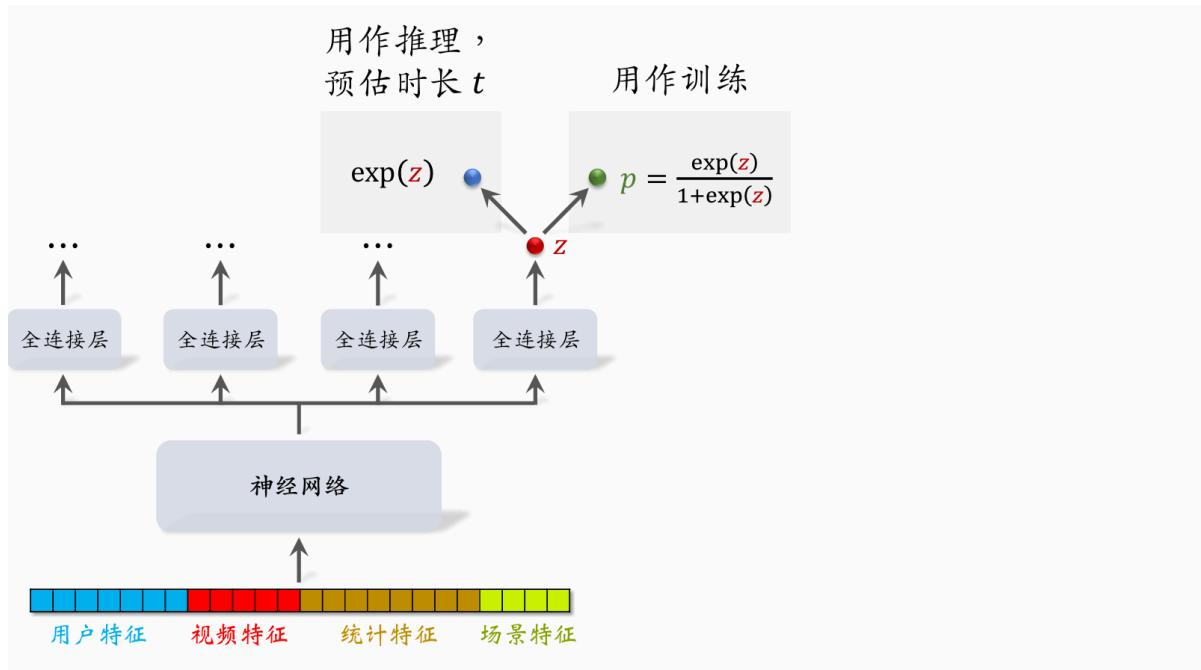
视频播放时长

图文 vs 视频

- 图文笔记排序的主要依据：
点击、点赞、收藏、转发、评论……
- 视频排序的依据还有播放时长和完播。
- 直接用回归拟合播放时长效果不好。建议用 YouTube 的时长建模。



如果 $p = y$, 那么 $\exp(z) = t$.



视频播放时长建模

- 把最后一个全连接层的输出记作 z 。设 $p = \text{sigmoid}(z)$ 。
- 实际观测的播放时长记作 t 。（如果没有点击，则 $t = 0$ 。）
- 做训练：最小化交叉熵损失

$$-\left(\frac{t}{1+t} \cdot \log p + \frac{1}{1+t} \cdot \log(1-p) \right)。$$

- 做推理：把 $\exp(z)$ 作为播放时长的预估。
- 把 $\exp(z)$ 作为融合公式中的一项。

视频完播

回归方法

- 例：视频长度 10 分钟，实际播放 4 分钟，则实际播放率为 $y = 0.4$ 。
- 让预估播放率 p 拟合 y :

$$\text{loss} = y \cdot \log p + (1 - y) \cdot \log(1 - p)。$$

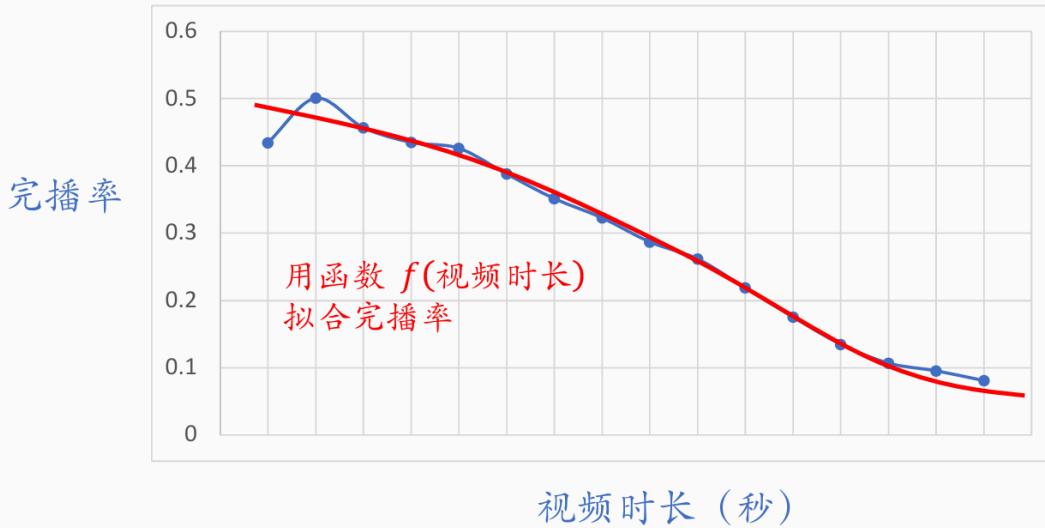
- 线上预估完播率，模型输出 $p = 0.73$ ，意思是预计播放 73%

二元分类方法

- 定义完播指标，比如完播 80%。
- 例：视频长度 10 分钟，播放 >8 分钟作为正样本，播放 <8 分钟作为负样本。
- 做二元分类训练模型：播放 >80% vs 播放 <80%。
- 线上预估完播率，模型输出 $p = 0.73$ ，意思是：

$$\mathbb{P}(\text{播放} > 80\%) = 0.73。$$

不能直接把预估的完播率用到融分公式 (why ?)



视频越长完播率越低，直接用预估完播率会偏向推荐短视频而非长视频。因此需要进行优化，使短视频和长视频一样公平。

- 线上预估完播率，然后做调整：

$$p_{\text{finish}} = \frac{\text{预估完播率}}{f(\text{视频长度})}$$

- 把 p_{finish} 作为融合公式中的一项。

排序模型的特征

特征

用户画像 (User Profile)

- 用户 ID (在召回、排序中做 embedding)。
- 人口统计学属性：性别、年龄。
- 账号信息：新老、活跃度……
- 感兴趣的类目、关键词、品牌。

物品画像 (Item Profile)

- 物品 ID (在召回、排序中做 embedding)。
- 发布时间 (或者年齿)。
- GeoHash (经纬度编码)、所在城市。
- 标题、类目、关键词、品牌……
- 字数、图片数、视频清晰度、标签数……
- 内容信息量、图片美学……

用户统计特征

- 用户最近 30 天 (7 天、1 天、1 小时) 的曝光数、点击数、点赞数、收藏数……
- 按照笔记 图文/视频分桶。 (比如最近 7 天，该用户对图文笔记的点击率、对视频笔记的点击率。)

- 按照笔记类目分桶。（比如最近 30 天，用户对美妆笔记的点击率、对美食笔记的点击率、对科技数码笔记的点击率。）

笔记统计特征

- 笔记最近 30 天（7 天、1 天、1 小时）的曝光数、点击数、点赞数、收藏数……
- 按照用户性别分桶、按照用户年龄分桶……
- 作者特征：
 - 发布笔记数
 - 粉丝数
 - 消费指标（曝光数、点击数、点赞数、收藏数）

场景特征 (Context)

- 用户定位 GeoHash（经纬度编码）、城市。
- 当前时刻（分段，做 embedding）。
- 是否是周末、是否是节假日。
- 手机品牌、手机型号、操作系统。

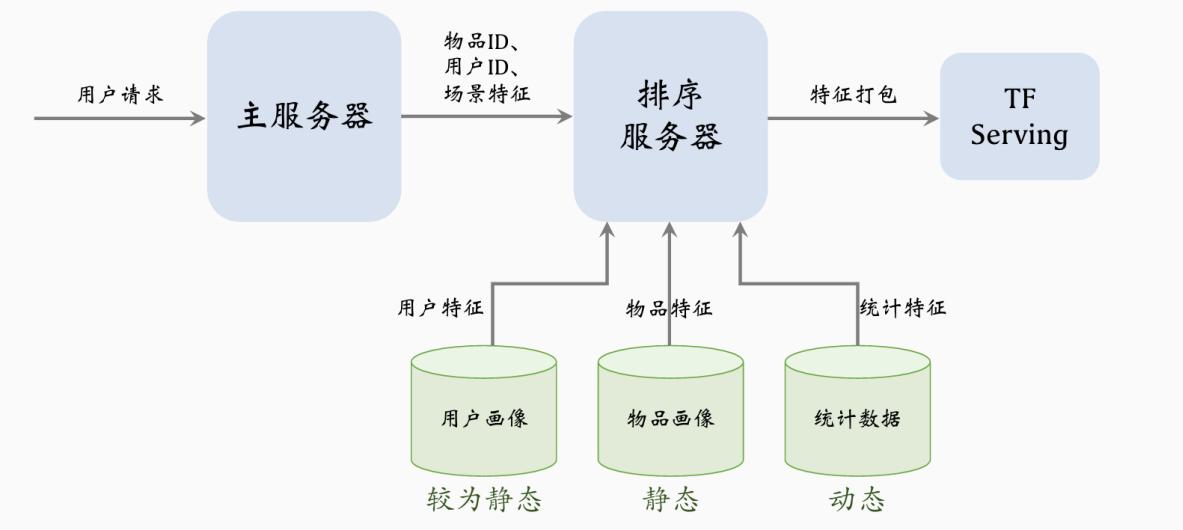
特征处理

- **离散特征**：做 embedding。
 - 用户 ID、笔记 ID、作者 ID。
 - 类目、关键词、城市、手机品牌。
- **连续特征**：做分桶，变成离散特征。
 - 年龄、笔记字数、视频长度。
- **连续特征**：其他变换。
 - 曝光数、点击数、点赞数等数值做 $\log(1 + x)$ 。
 - 转化为点击率、点赞率等值，并做平滑。

特征覆盖率

- 很多特征无法覆盖 100% 样本。
- 例：很多用户不填年龄，因此用户年龄特征的覆盖率远小于 100%。
- 例：很多用户设置隐私权限，APP 不能获取用户地理定位，因此场景特征有缺失。
- 提高特征覆盖率，可以让精排模型更准。

数据服务



主服务器从召回服务器召回一批物品ID。用户发送请求，主服务器将用户ID，场景特征以及物品ID发送给排序服务器。

用户画像数据库压力较小，因为每次只读一个用户的特征。物品画像压力比较大，因为每次要读几千篇笔记的特征。同理，用户统计值数据库压力较小，物品统计值数据压力很大。用户画像的向量可以很大，但是物品画像的向量不要很大。

用户画像以及物品画像都比较静态，甚至可以将用户画像和物品画像直接缓存在排序服务器本地。统计数据是动态的，需要及时更新数据库。

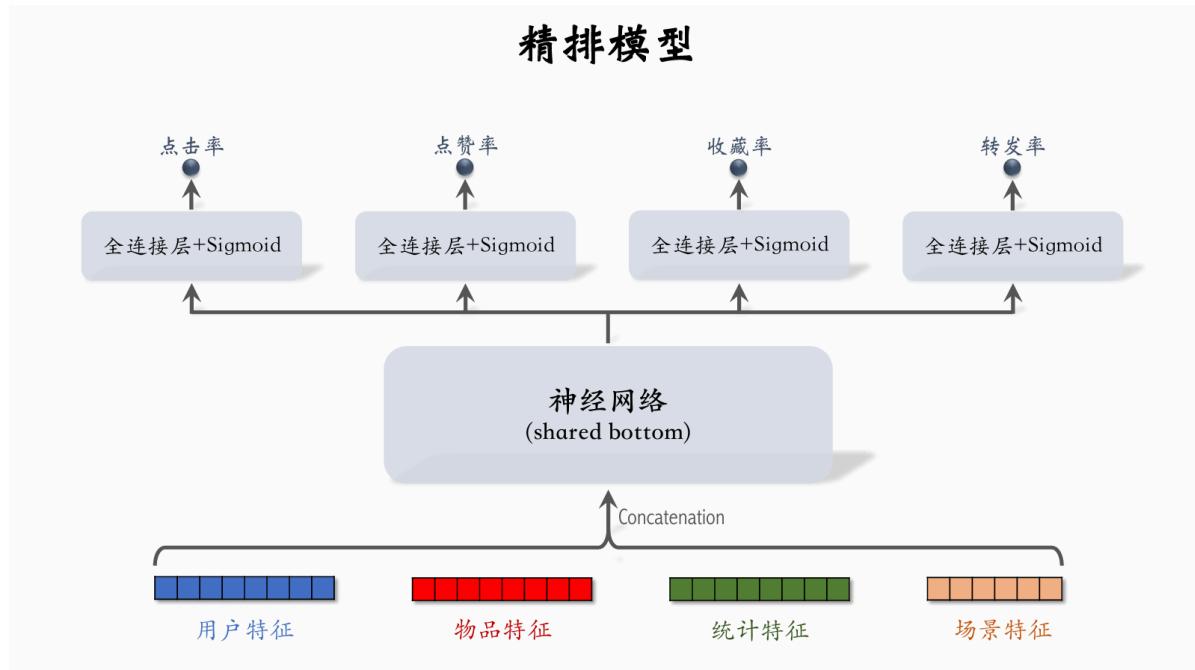
排序服务器在收取到特征后，将特征打包给 TF Serving 。TF 会给笔记打分，将结果返回给排序服务器。排序服务器根据一系列规则给笔记排序，把排名最高的几十篇笔记返回给服务器。

粗排

粗排 VS 精排

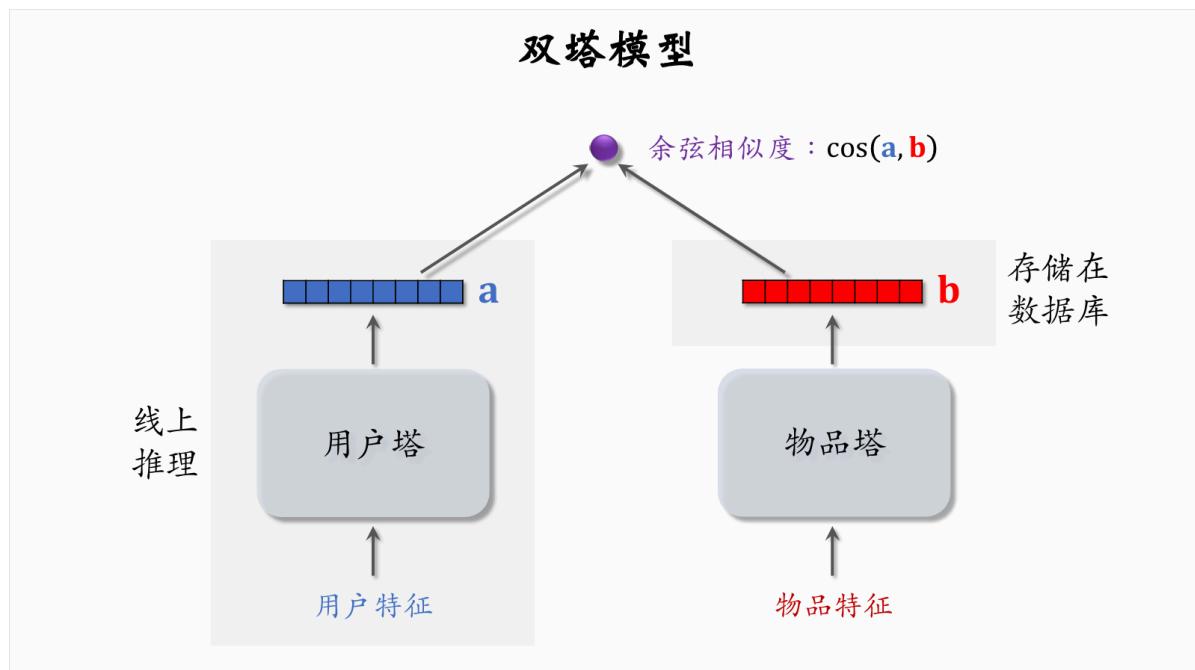
粗排	精排
给几千篇笔记打分。	给几百篇笔记打分。
单次推理代价必须小。	单次推理代价很大。
预估的准确性不高。	预估的准确性更高。

精排模型&双塔模型



精排模型

- 前期融合：先对所有特征做 concatenation，再输入神经网络。
- 线上推理代价大：如果有 n 篇候选笔记，整个大模型要做 n 次推理。



双塔模型

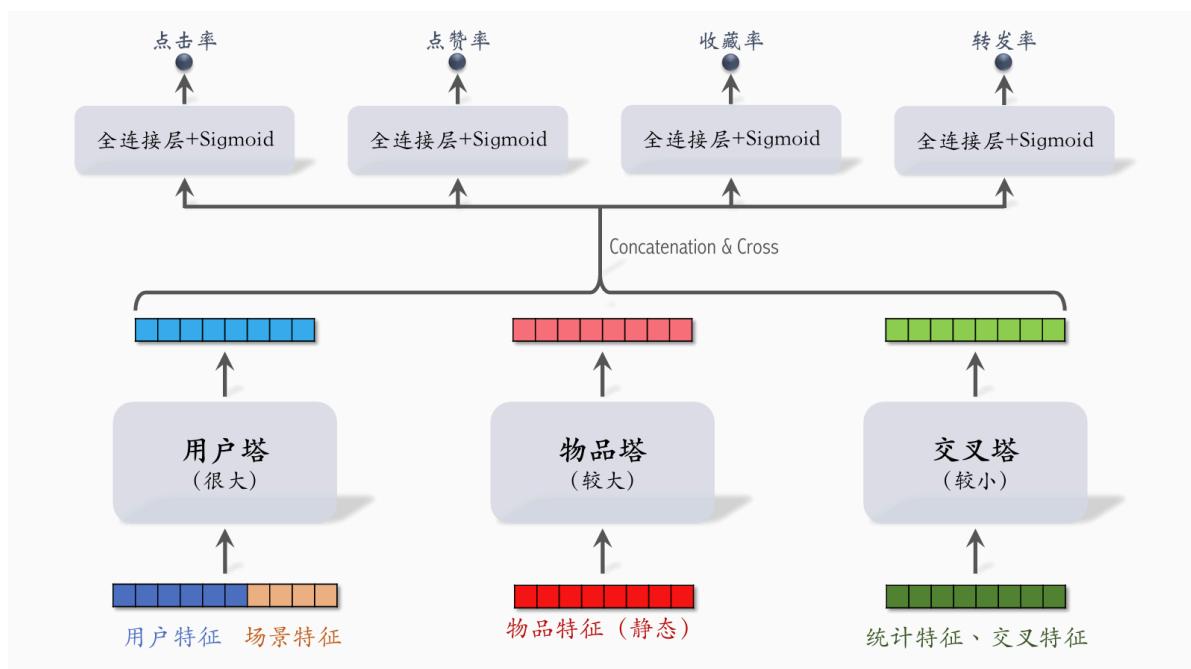
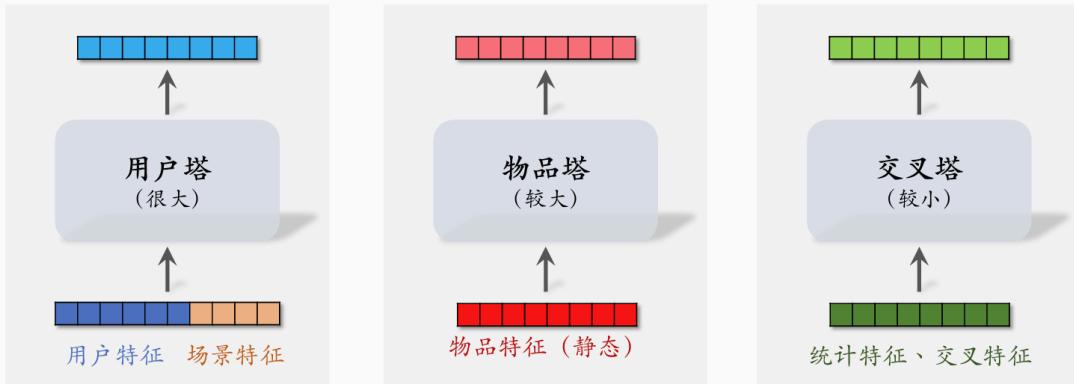
- 后期融合：把用户、物品特征分别输入不同的神经网络，不对用户、物品特征做融合。
- 线上计算量小：
 - 用户塔只需要做一次线上推理，计算用户表征 a 。
 - 物品表征 b 事先储存在向量数据库中，物品塔在线上不做推理。
- 预估准确性不如精排模型。

后期融合准确性不如前期融合

粗排的三塔模型

粗排的三塔模型

- 只有一个用户，用户塔只做一次推理。
- 即使用户塔很大，总计算量也不大。
- 有 n 个物品，理论上物品塔需要做 n 次推理。
- PS 缓存物品塔的输出向量，避免绝大部分推理。
- 统计特征动态变化，缓存不可行。
- 有 n 个物品，交叉塔必须做 n 次推理。



- 有 n 个物品，模型上层需要做 n 次推理。
- 粗排推理的大部分计算量在模型上层。

三塔模型的推理

- 从多个数据源取特征：
 - 1 个用户的画像、统计特征。
 - n 个物品的画像、统计特征。
- 用户塔：只做 1 次推理。
- 物品塔：未命中缓存时需要做推理。
- 交叉塔：必须做 n 次推理。
- 上层网络：做 n 次推理，给 n 个物品打分。

特征交叉

Factorized Machine (FM)

线性模型

- 有 d 个特征，记作 $\mathbf{x} = [x_1, \dots, x_d]$ 。
- 线性模型：

$$p = b + \sum_{i=1}^d w_i x_i.$$

- 模型有 $d + 1$ 个参数： $\mathbf{w} = [w_1, \dots, w_d]$ 和 b 。
- 预测是特征的加权和。（只有加，没有乘。）

二阶交叉特征

- 有 d 个特征，记作 $\mathbf{x} = [x_1, \dots, x_d]$ 。
- 线性模型 + 二阶交叉特征：

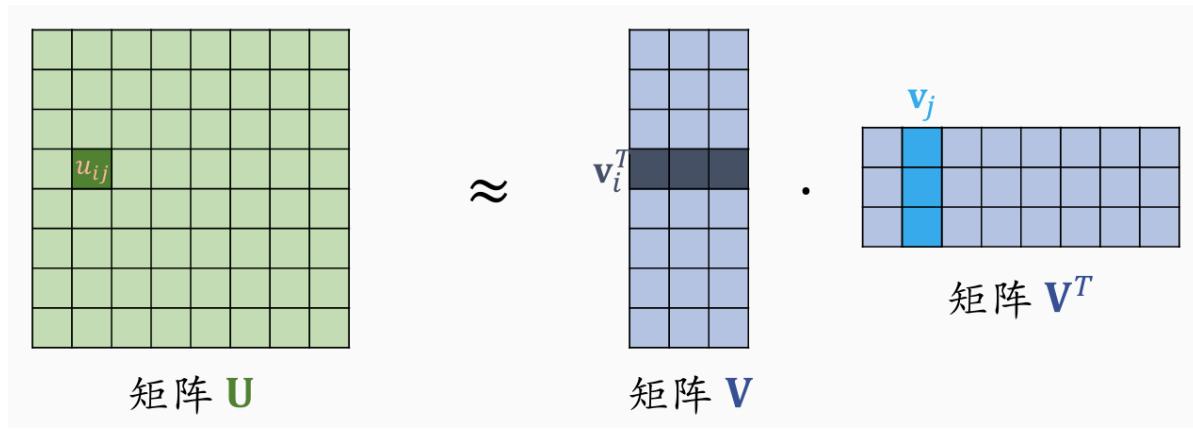
$$p = b + \sum_{i=1}^d w_i x_i + \sum_{i=1}^d \sum_{j=i+1}^d u_{ij} x_i x_j.$$

- 模型有 $O(d^2)$ 个参数。

线性模型 + 二阶交叉特征：

$$p = b + \sum_{i=1}^d w_i x_i + \sum_{i=1}^d \sum_{j=i+1}^d u_{ij} x_i x_j.$$

$$u_{ij} \approx v_i^T v_j$$



矩阵 U d 行 d 列，矩阵 V d 行 k 列，矩阵 V^T k 行 d 列。

- Factorized Machine (FM)：

$$p = b + \sum_{i=1}^d w_i x_i + \sum_{i=1}^d \sum_{j=i+1}^d (\mathbf{v}_i^T \mathbf{v}_j) x_i x_j.$$

- FM 模型有 $O(kd)$ 个参数。 $(k \ll d)$

Factorized Machine

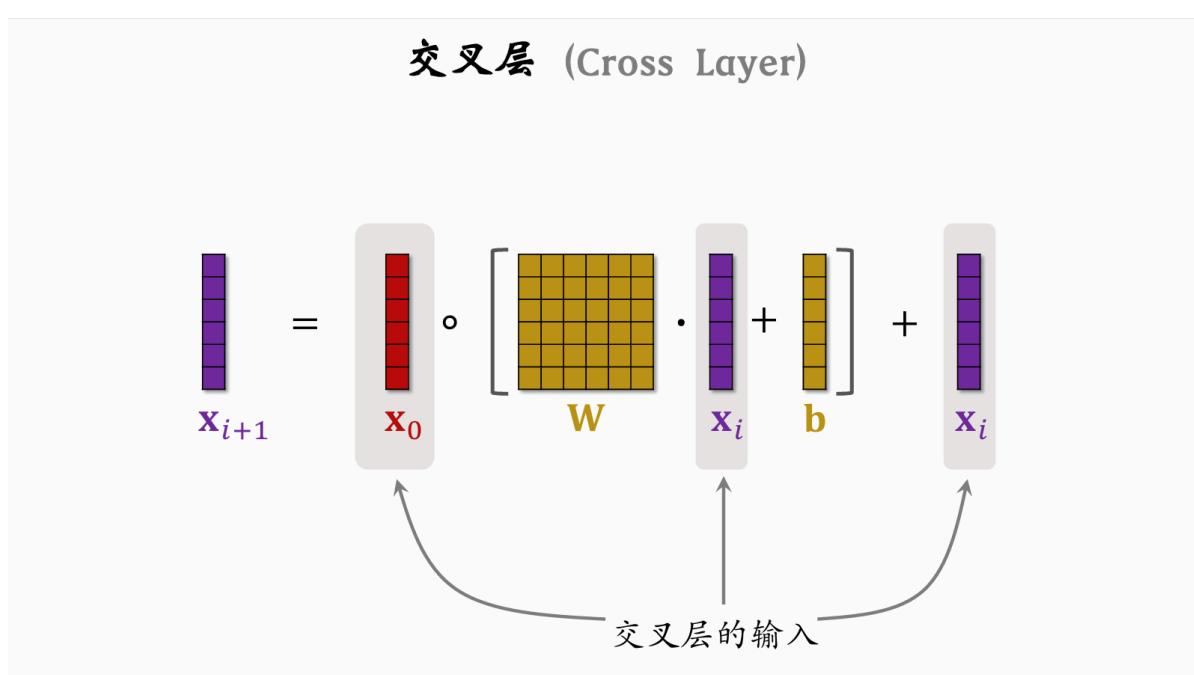
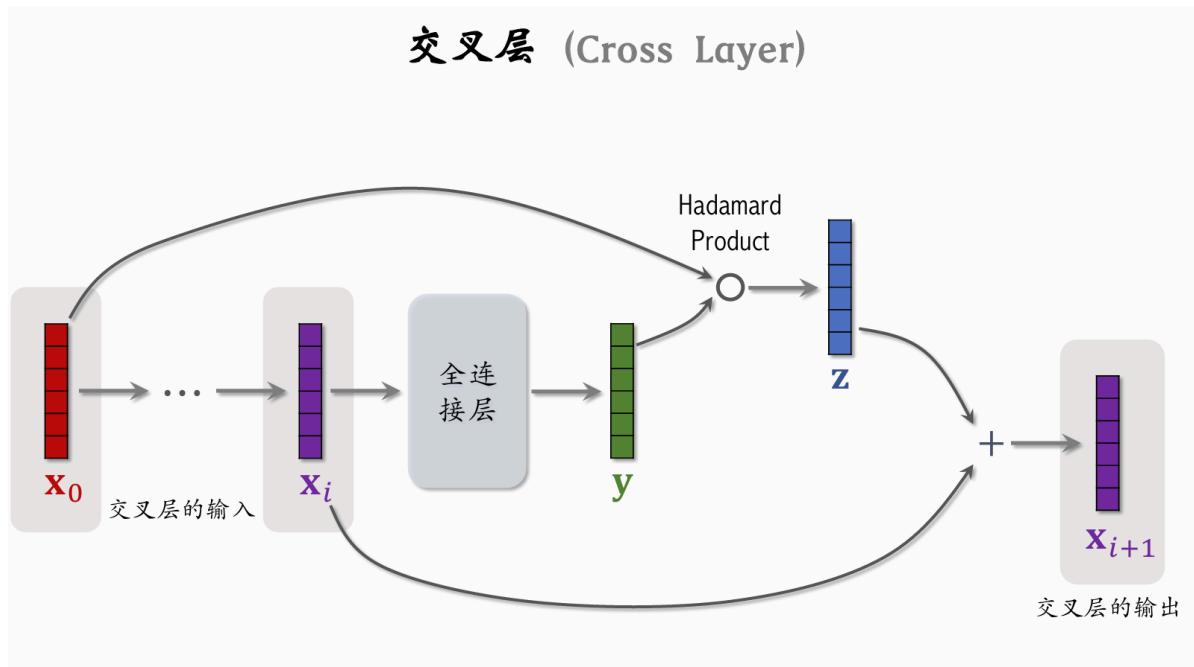
- FM 是线性模型的替代品，能用线性回归、逻辑回归的场景，都可以用 FM。
- FM 使用二阶交叉特征，表达能力比线性模型更强。
- 通过做近似 $u_{ij} \approx \mathbf{v}_i^T \mathbf{v}_j$ ，FM 把二阶交叉权重的数量从 $O(d^2)$ 降低到 $O(kd)$ 。

深度交叉网络 (DCN)

召回、排序模型

双塔模型和多目标排序模型只是结构，内部的神经网络可以用任意网络。

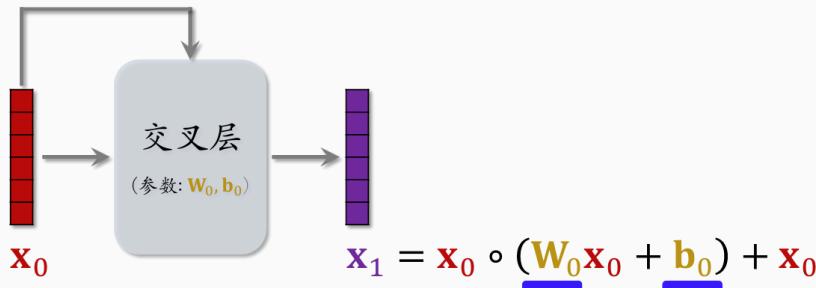
交叉层 (Cross Layer)



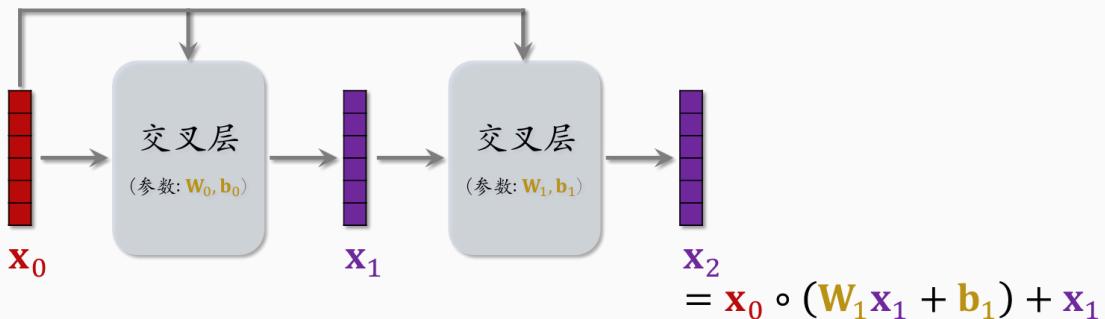
利用 Resnet 思想，防止梯度消失

交叉网络 (Cross Network)

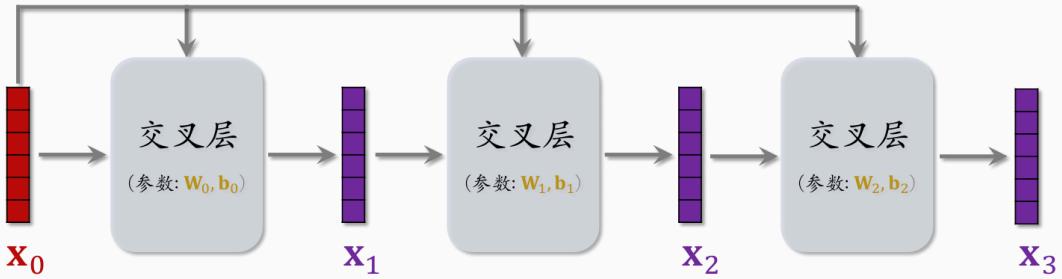
交叉网络 (Cross Network)



交叉网络 (Cross Network)

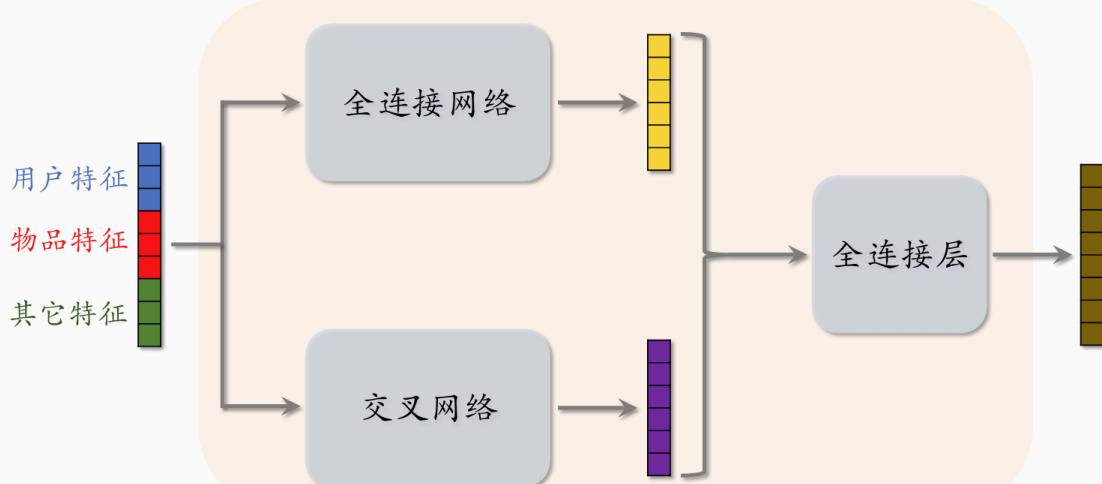


交叉网络 (Cross Network)



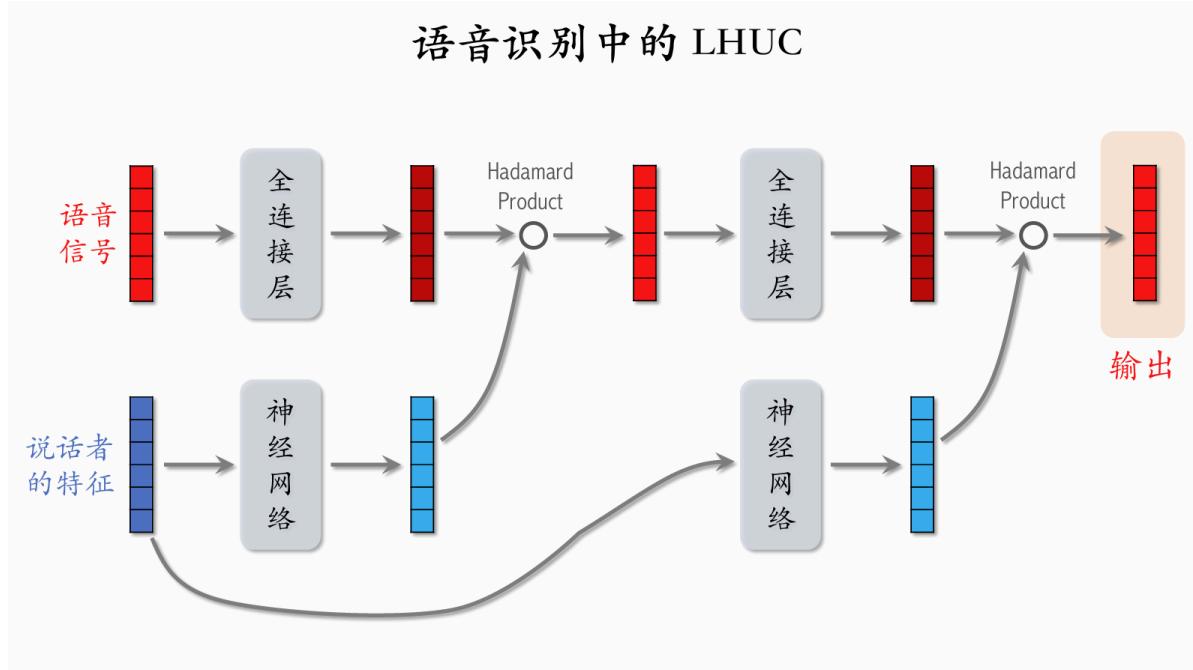
深度交叉网络 (Deep & Cross Network)

深度交叉网络 (Deep & Cross Network)

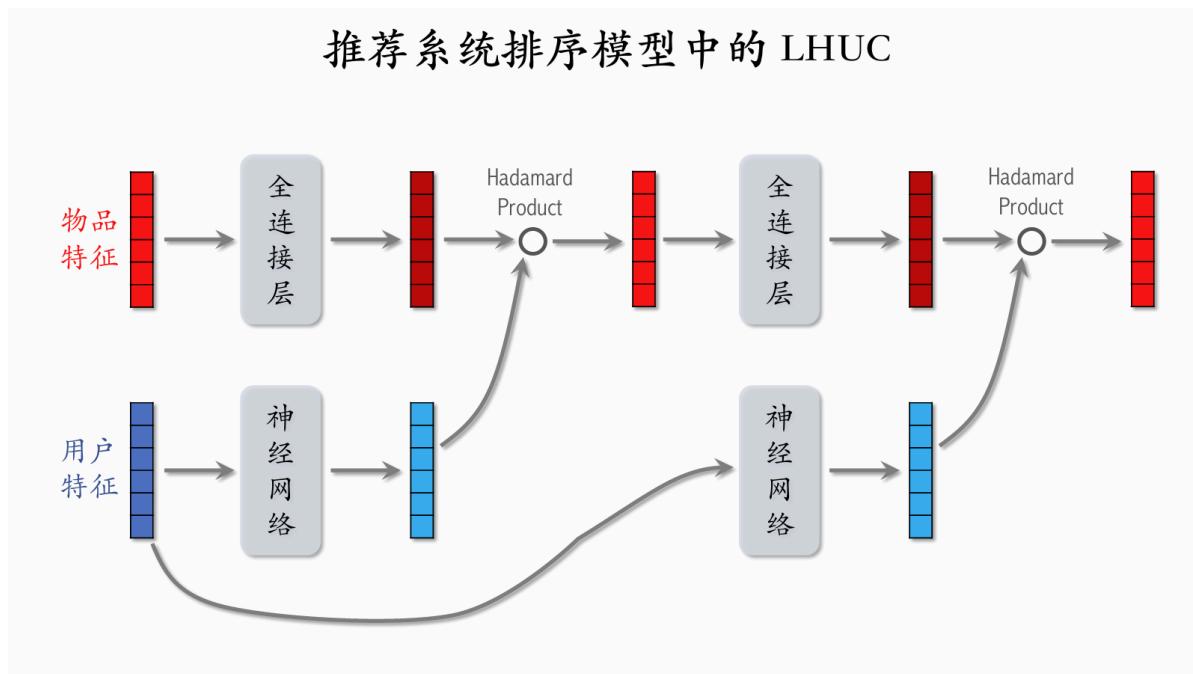


DCN 的实际效果优于全连接，可以用于双塔模型中的用户塔和物品塔，多目标排序模型中的 shared bottom 神经网络，以及MMoE中的专家神经网络。

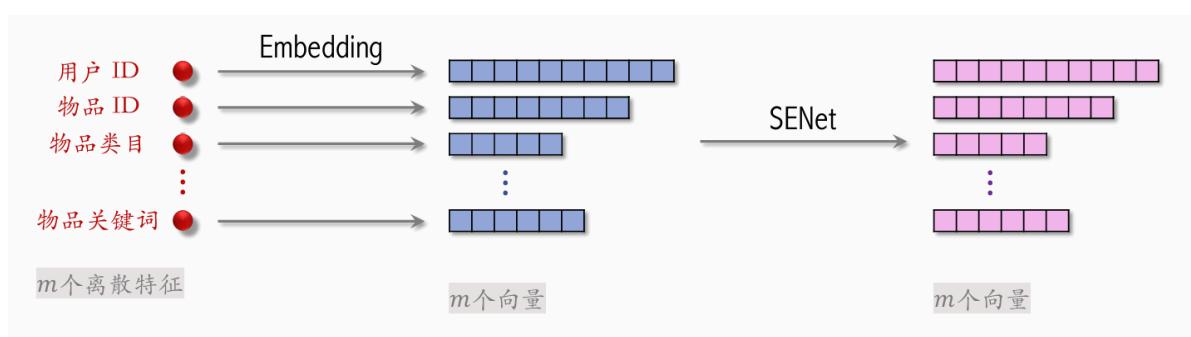
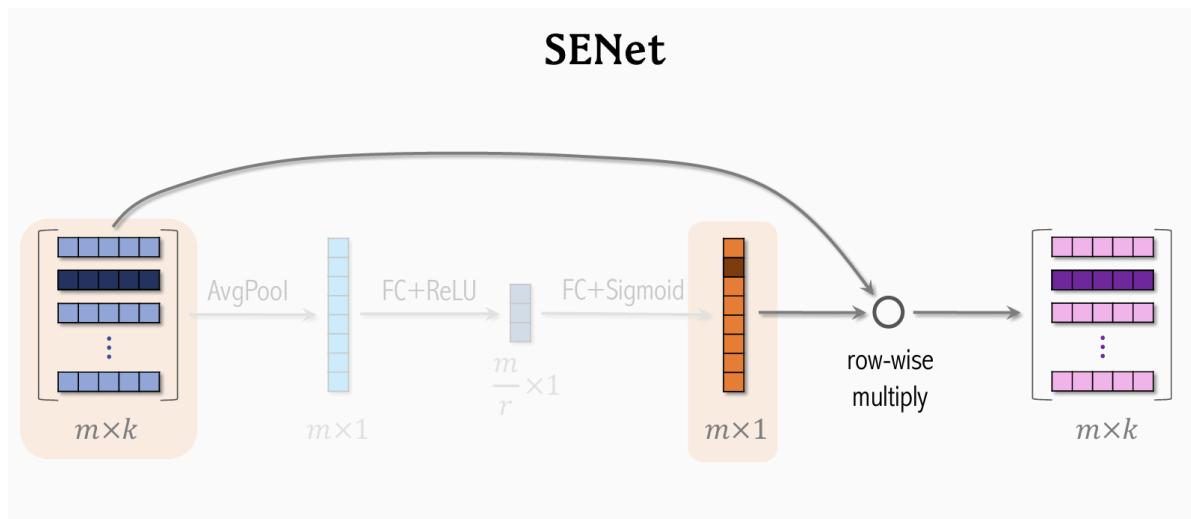
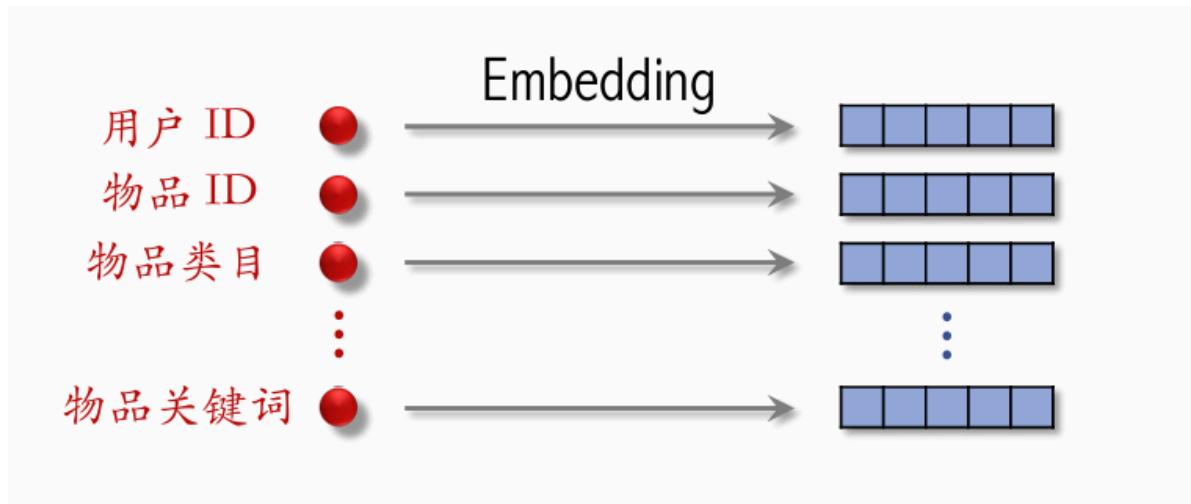
Learning Hidden Unit Contributions (LHUC)



神经网络中的结构为[多个全连接层] → [Sigmoid 乘以 2]，这样神经网络的输出向量中都是 0 到 2 之间的数。

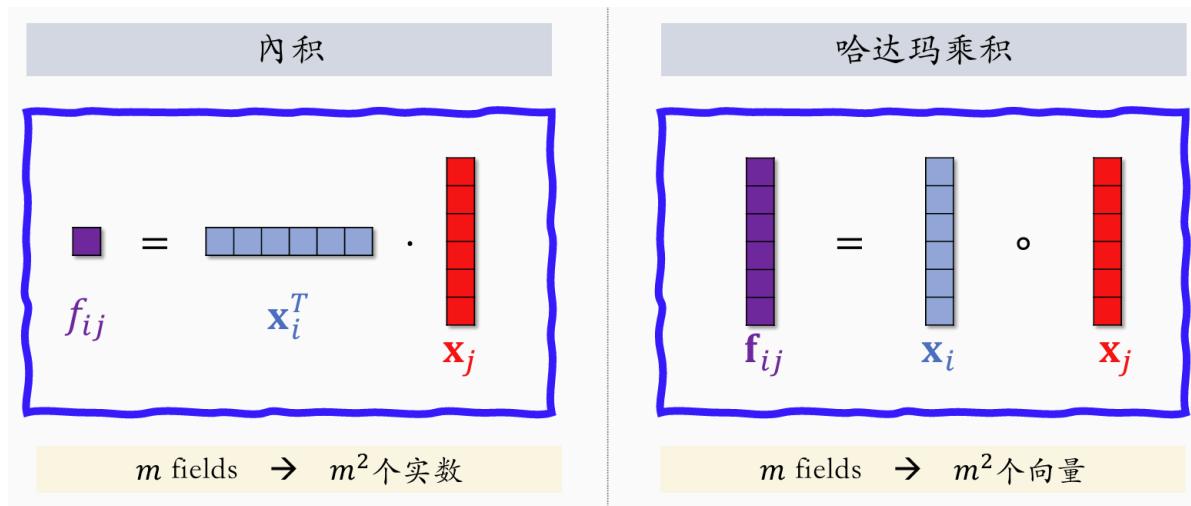


SENet & Bilinear Cross



- **SENet 对离散特征做 field-wise 加权。**
- **Field :**
 - 用户 ID Embedding 是 64 维向量。
 - 64 个元素 (即一个特征的 embedding 向量) 算一个 field, 获得相同的权重。
 - 特征越重要, 获得的权重越大。
- **如果有 m 个 fields, 那么权重向量是 m 维。**

Field 间特征交叉

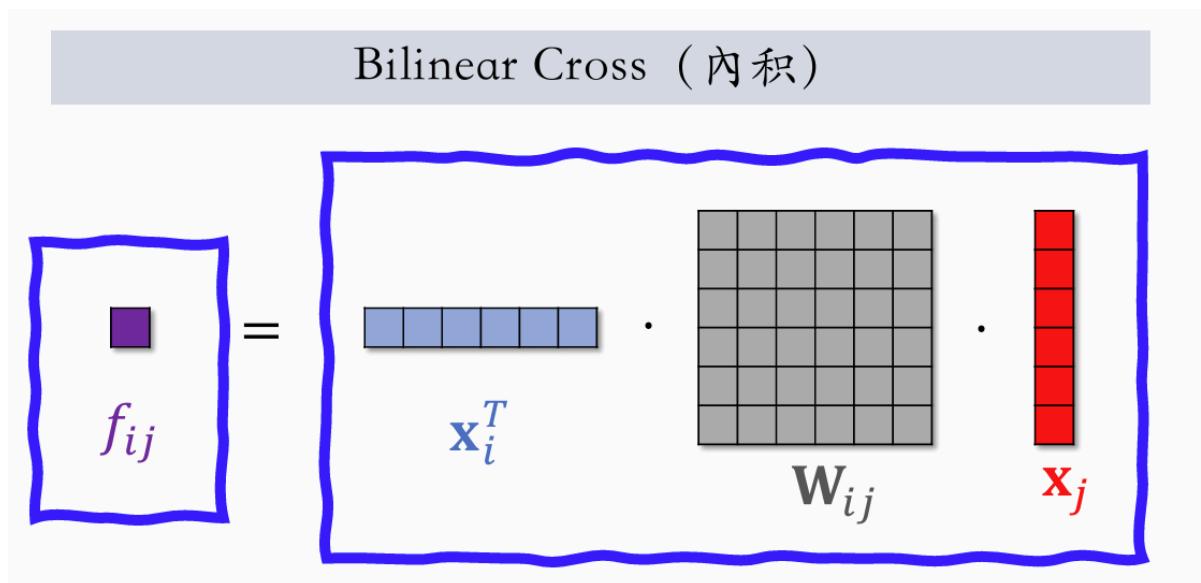


内积

x_i^T 和 x_j 都是特征的 embedding 向量, f_{ij} 是一个实数, 如果有 m 个 field, 他们之间两两内积, 就会有 m^2 个实数。

哈达玛乘积

x_i^T 和 x_j 都是特征的 embedding 向量, f_{ij} 是一个向量, 如果有 m 个 field, 他们之间两两哈达玛乘积, 就会有 m^2 个向量。量太大, 需要人工指定一部分向量做交叉, 而不是所有向量都交叉。



Bilineard Cross (内积)

如果有 m 个 field, 就会有 m^2 个实数 f_{ij} , $m^2/2$ 个参数矩阵 W_{ij} 。

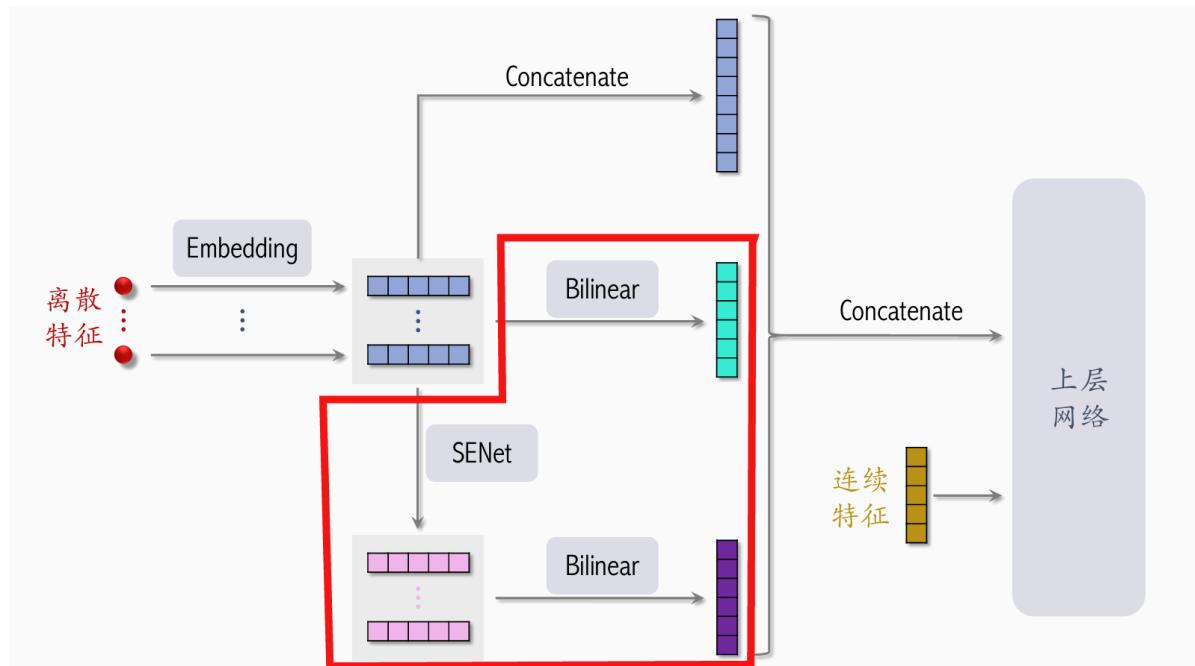
Bilinear Cross (哈达玛乘积)

$$\mathbf{f}_{ij} = \mathbf{x}_i \circ \mathbf{W}_{ij} \mathbf{x}_j$$

Bilinear Cross (哈达玛)

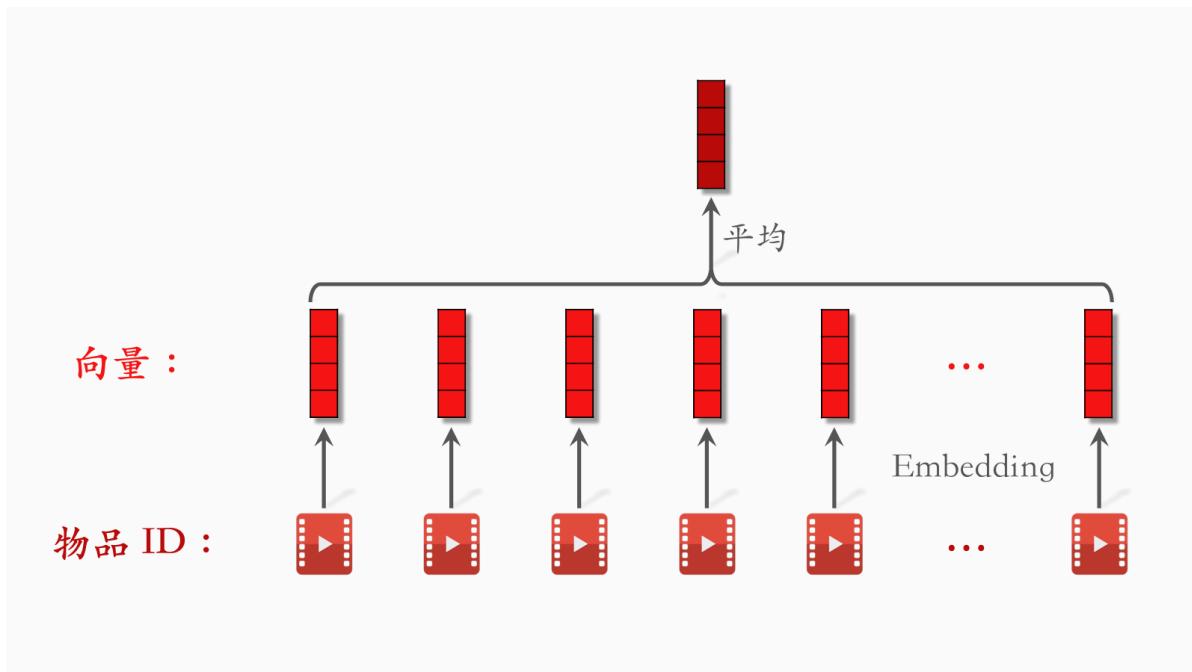
如果有 m 个 field，就会有 m^2 个向量 \mathbf{f}_{ij} , $m^2/2$ 个参数矩阵 \mathbf{W}_{ij} 。

FiBiNet



行为序列

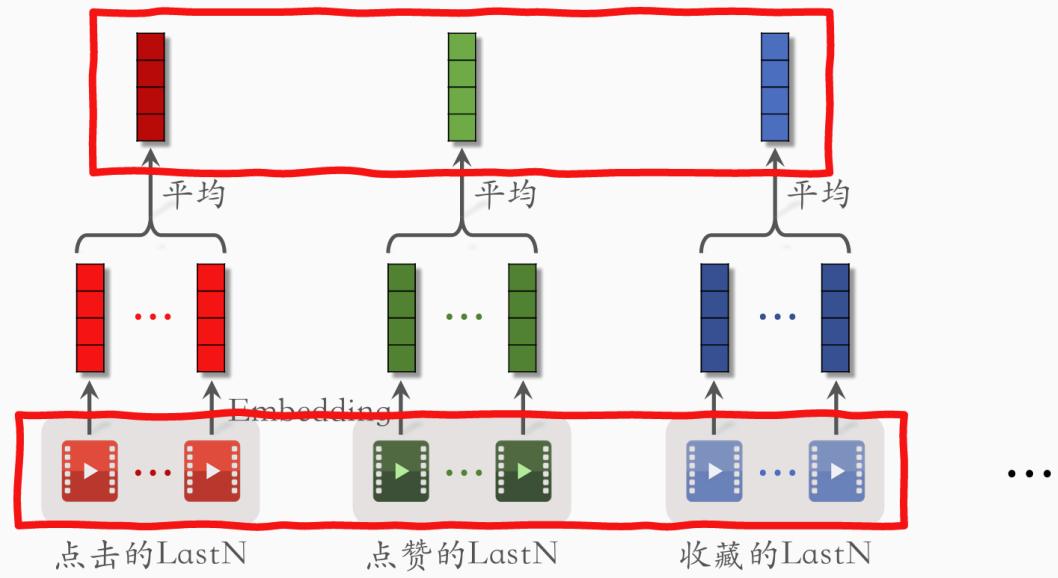
用户行为序列建模



LastN 特征

- **LastN**: 用户最近的 n 次交互（点击、点赞等）的物品 ID。
- 对 **LastN** 物品 ID 做 embedding，得到 n 个向量。
- 把 n 个向量取平均，作为用户的一种特征。
- 适用于召回双塔模型、粗排三塔模型、精排模型。

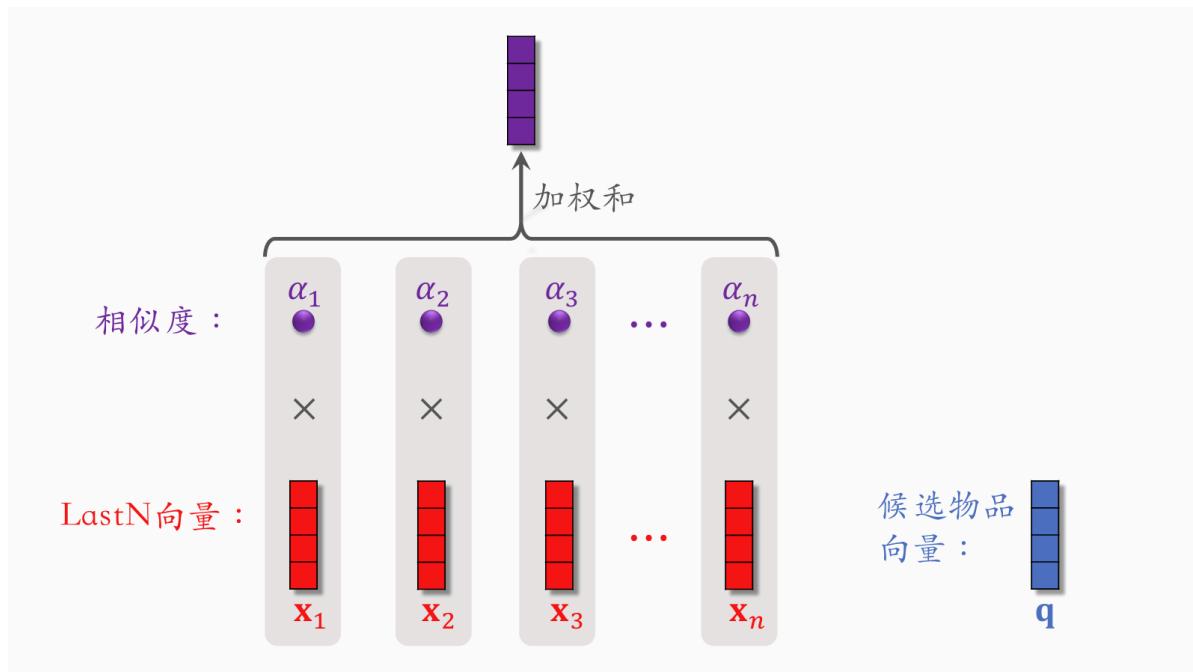
小红书的实践



DIN 模型

DIN 模型

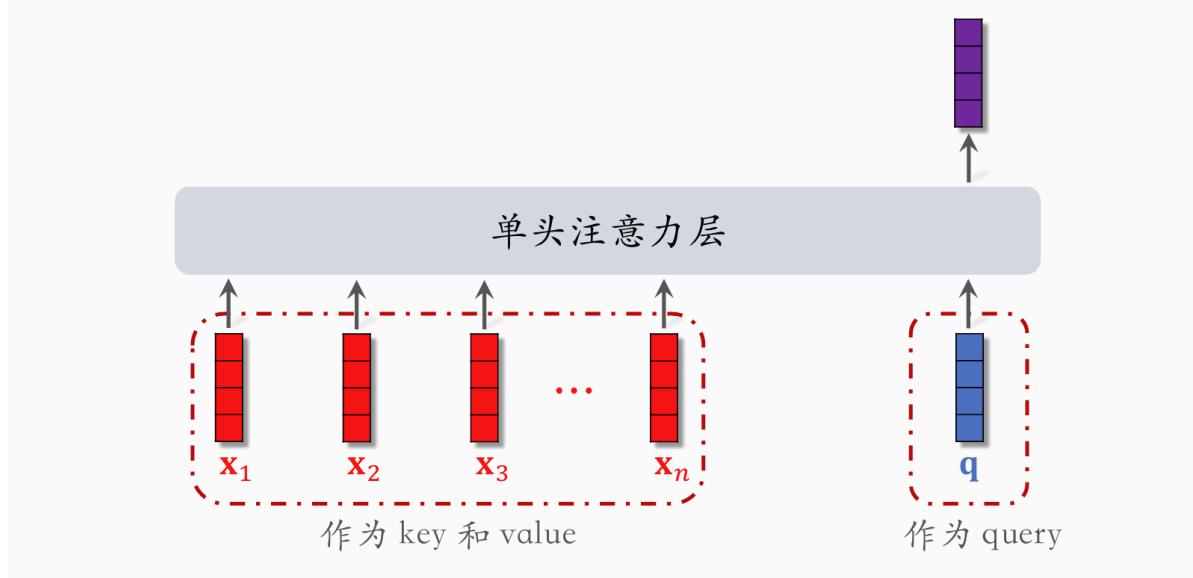
- DIN 用 加权平均 代替 平均，即注意力机制（attention）。
- 权重：候选物品与用户 **LastN** 物品的相似度。



DIN 模型

- 对于某候选物品，计算它与用户 **LastN** 物品的相似度。
- 以相似度为权重，求用户 **LastN** 物品向量的加权和，结果是一个向量。
- 把得到的向量作为一种用户特征，输入排序模型，预估（用户，候选物品）的点击率、点赞率等指标。
- 本质是注意力机制（attention）。

DIN的本质是注意力机制



简单平均 v.s 注意力机制

- 简单平均和注意力机制都适用于精排模型。
- 简单平均适用于双塔模型、三塔模型。
 - 简单平均只需要用到 **LastN**，属于用户自身的特征。
 - 把 LastN 向量的平均作为用户塔的输入。
- 注意力机制不适用于双塔模型、三塔模型。
 - 注意力机制需要用到 **LastN + 候选物品**。

- 用户塔看不到候选物品，不能把注意力机制用在用户塔。

SIM模型

DIN 模型

- 计算用户 **LastN** 向量的加权平均。
- 权重是候选物品与 LastN 物品的相似度。

DIN 模型的缺点

- 注意力层的计算量 $\propto n$ (用户行为序列的长度)。
- 只能记录最近几百个物品，否则计算量太大。
- 缺点：关注短期兴趣，遗忘长期兴趣。

如何改进 DIN？

- **目标**：保留用户长期行为序列 (n 很大)，而且计算量不会过大。
- **改进 DIN**：
 - DIN 对 **LastN** 向量做加权平均，权重是相似度。
 - 如果某 **LastN** 物品与候选物品差异很大，则权重接近零。
 - 快速排除掉与候选物品无关的 **LastN** 物品，降低注意力层的计算量。

SIM 模型

- 保留用户长期行为记录， n 的大小可以是几千。
- 对于每个候选物品，在用户 **LastN** 记录中做快速查找，找到 k 个相似物品。
- 把 **LastN** 变成 **TopK**，然后输入到注意力层。
- **SIM** 模型减小计算量 (从 n 降到 k)。

第一步：查找

- **方法一：Hard Search**
 - 根据候选物品的类别，保留 **LastN** 物品中类别相同的。
 - 简单，快速，无需训练。
- **方法二：Soft Search**
 - 把物品做 **embedding**，变成向量。
 - 把候选物品向量作为 **query**，做 k 近邻查找，保留 **LastN** 物品中最近的 k 个。
 - 效果更好，编程实现更复杂。

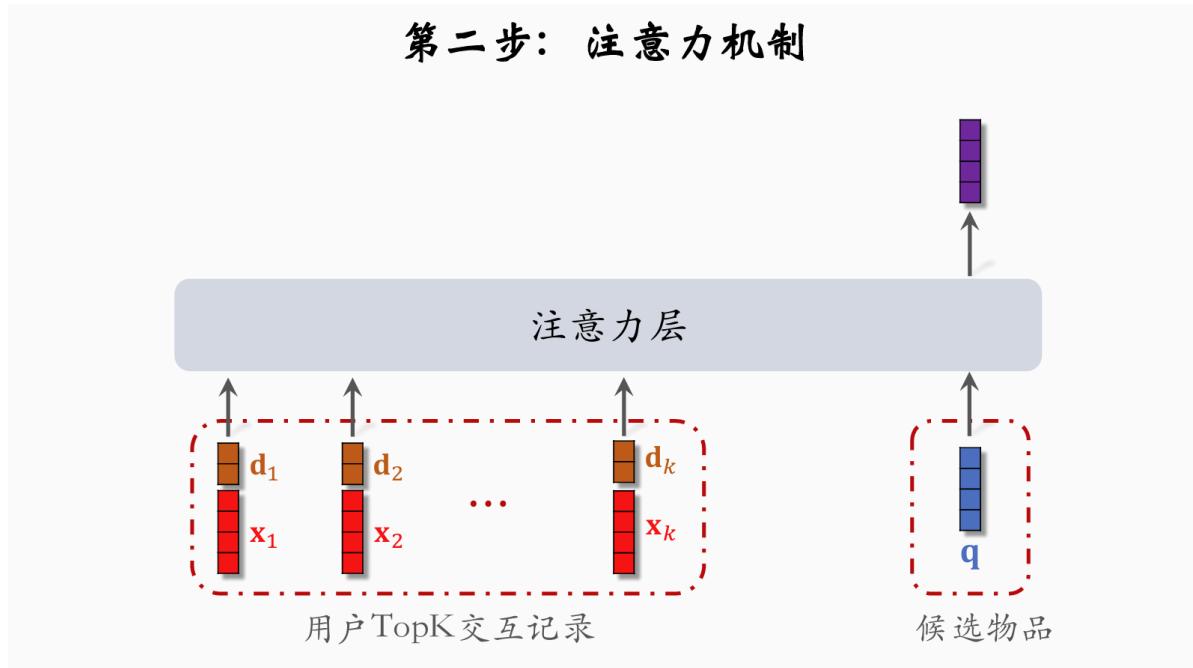
第二步：注意力机制

使用时间信息

- 用户与某个 **LastN** 物品的交互时刻距离今为 δ 。
- 对 δ 做离散化，再做 **embedding**，变成向量 d 。
- 把两个向量做 **concatenation**，表征一个 **LastN** 物品。
 - 向量 x 是物品 **embedding**。

- 向量 d 是时间的 **embedding**。

第二步：注意力机制



为什么 SIM 使用时间信息？

- DIN 的序列短，记录用户近期行为。
- SIM 的序列长，记录用户长期行为。
- 时间越久远，重要性越低。

结论

- 长序列（长期兴趣）优于短序列（近期兴趣）。
- 注意力机制优于简单平均。
- Soft search 还是 Hard search？取决于工程基建。
- 使用时间信息有提升。

重排

推荐系统中的多样性

物品相似性的度量

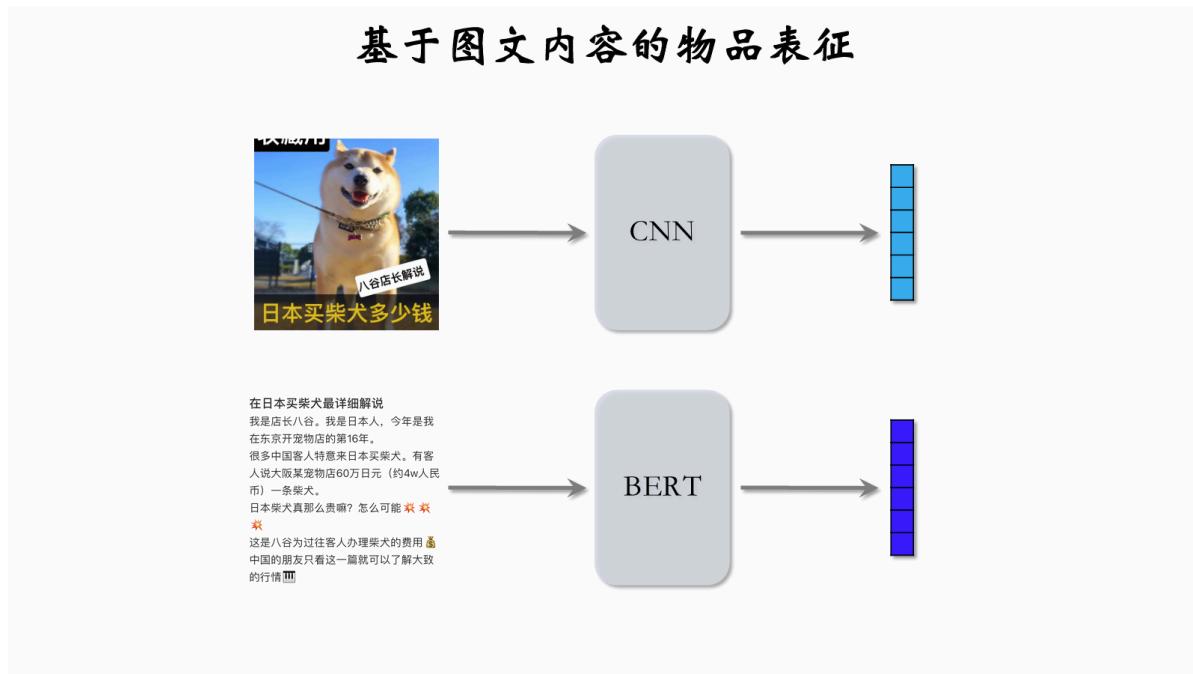
相似性的度量

- 基于物品属性标签。
 - 类目、品牌、关键词.....
- 基于物品向量表征。
 - 用召回的双塔模型学到的物品向量（不好）。
 - 基于内容的向量表征（好）。

基于物品属性标签

- 物品属性标签：类目、品牌、关键词.....
- 根据 一级类目、二级类目、品牌 计算相似度。
 - 物品 i ：美妆、彩妆、香奈儿。

- 物品 j : 美妆、香水、香奈儿。
- 相似度: $\text{sim}_1(i, j) = 1$, $\text{sim}_2(i, j) = 0$, $\text{sim}_3(i, j) = 1$.



基于图文内容的物品表征可以提升多样性。

可以用 CNN 处理图片输出特征向量, BERT 处理文字输出特征向量。最后将两个向量拼起来即可。

如何训练 CNN 与 BERT?

- **CLIP** 是当前公认最有效的预训练方法。
- **思想**: 对于 图片—文本二元组, 预测图文是否匹配。
- **优势**: 无需人工标注。小红书的笔记天然包含图片 + 文字, 大部分笔记图文相关。

图片：



文字：



正样本

⋮

⋮



图片：



文字：



负样本

⋮

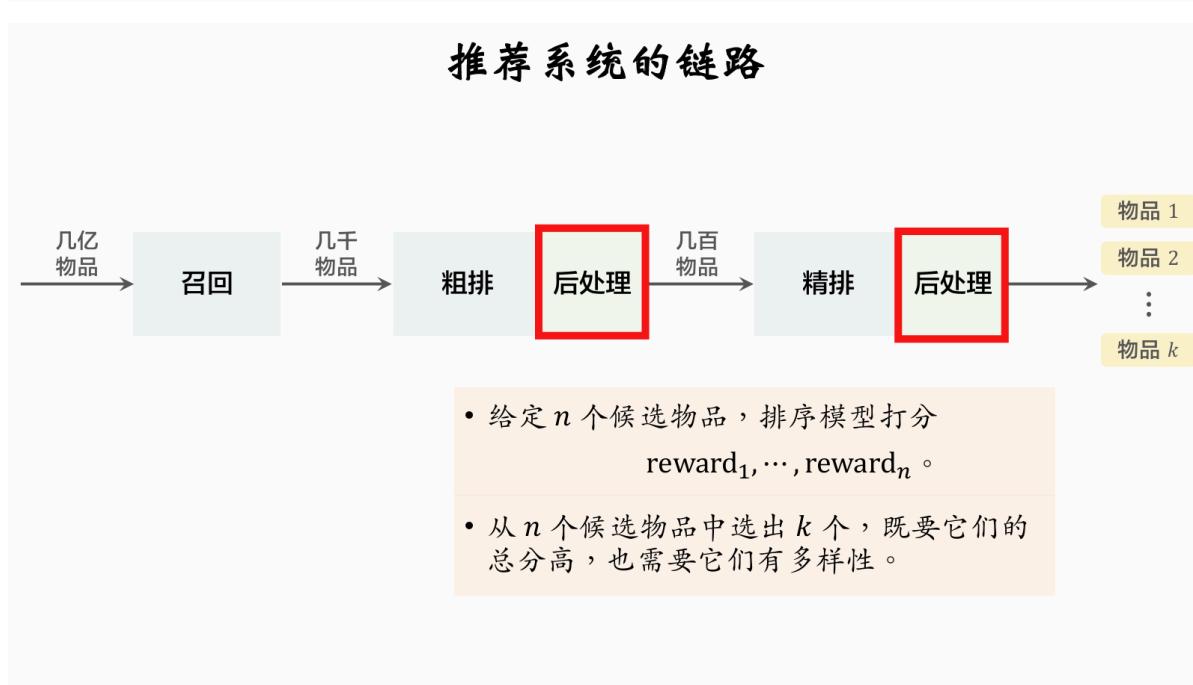
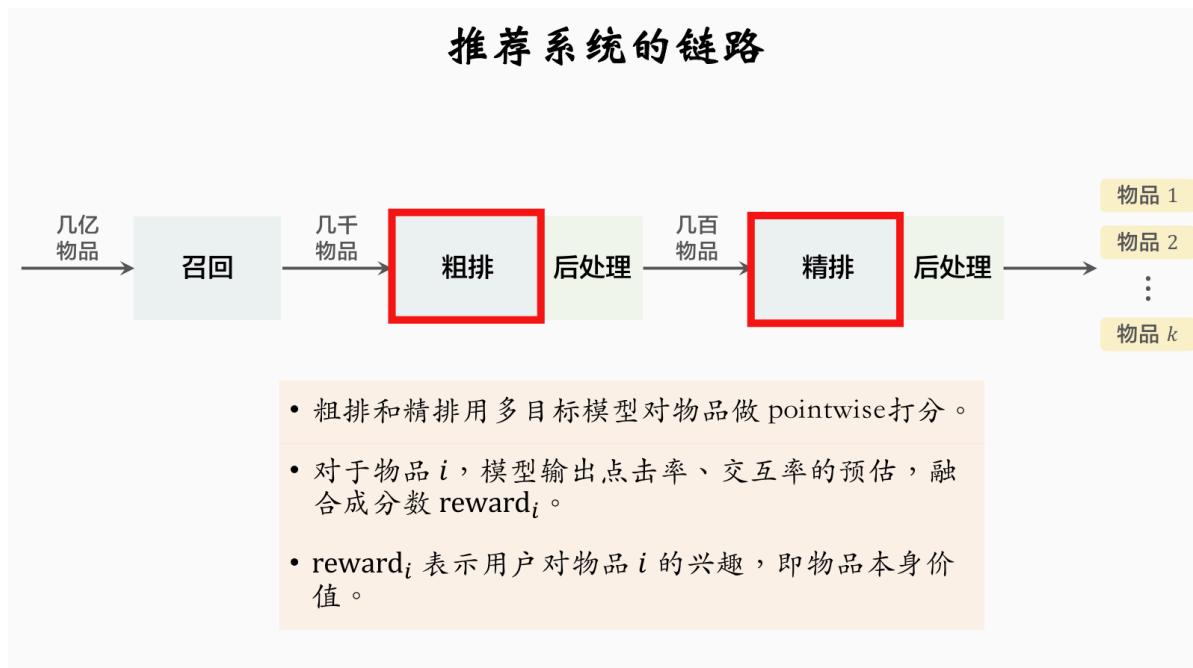
⋮



- 一个 batch 内有 m 对正样本。

- 一张图片和 $m - 1$ 条文本组成负样本。
 - 这个 batch 内一共有 $m(m - 1)$ 对负样本。

提升多样性的方法



粗排的后处理也需要多样性算法。

精排的后处理也被称为重排。

Maximal Marginal Relevance (MMR)

多样性

- 精排给 n 个候选物品打分，融合之后的分数为 $\text{reward}_1, \dots, \text{reward}_n$
 - 把第 i 和 j 个物品的相似度记作 $\text{sim}(i, j)$ 。
 - 从 n 个物品中选出 k 个，既要有高精排分数，也要有多样性。

MMR多样性算法



- 计算集合 \mathcal{R} 中每个物品 i 的 Marginal Relevance 分数:

$$\text{MR}_i = \theta \cdot \text{reward}_i - (1 - \theta) \cdot \max_{j \in \mathcal{S}} \text{sim}(i, j)$$

- reward_i 是物品的精排分数, $\max_{j \in \mathcal{S}} \text{sim}(i, j)$ 是未选中的物品 i 与已选中物品的相似程度。精排分数越高, 相似程度越低, 物品的 MR 分数就越高。
- Maximal Marginal Relevance (MMR):**
从未选中物品中选出 MR 分数最高的

$$\arg \max_{i \in \mathcal{R}} \text{MR}_i$$

MMR 多样性算法

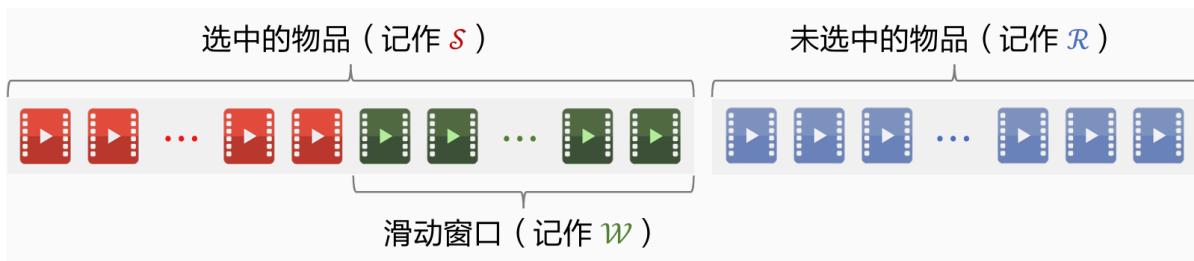
- 已选中的物品 \mathcal{S} 初始化为空集, 未选中的物品 \mathcal{R} 初始化为全集 $\{1, \dots, n\}$ 。
- 选择精排分数 reward_i 最高的物品, 从集合 \mathcal{R} 移到 \mathcal{S} 。
- 做 $k - 1$ 轮循环:
 - 计算集合 \mathcal{R} 中所有物品的分数 $\{\text{MR}_i\}_{i \in \mathcal{R}}$ 。
 - 选出分数最高的物品, 将其从 \mathcal{R} 移到 \mathcal{S} 。

滑动窗口

- MMR:**

$$\arg \max_{i \in \mathcal{R}} \left\{ \theta \cdot \text{reward}_i - (1 - \theta) \cdot \max_{j \in \mathcal{S}} \text{sim}(i, j) \right\}$$

- 已选中的物品越多 (即集合 \mathcal{S} 越大), 越难找出物品 $i \in \mathcal{R}$, 使得 i 与 \mathcal{S} 中的物品都不相似。
- 设 sim 的取值范围是 $[0, 1]$ 。当 \mathcal{S} 很大时, 多样性分数 $\max_{j \in \mathcal{S}} \text{sim}(i, j)$ 总是约等于 1, 导致 MMR 算法失效。
- 解决方案:** 设置一个滑动窗口 \mathcal{W} , 比如最近选中的 10 个物品, 用 \mathcal{W} 代替 MMR 公式中的 \mathcal{S} 。



- 标准 MMR:**

$$\arg \max_{i \in \mathcal{R}} \left\{ \theta \cdot \text{reward}_i - (1 - \theta) \cdot \max_{j \in \mathcal{S}} \text{sim}(i, j) \right\}.$$

- 用滑动窗口:**

$$\arg \max_{i \in \mathcal{R}} \left\{ \theta \cdot \text{reward}_i - (1 - \theta) \cdot \max_{j \in \mathcal{W}} \text{sim}(i, j) \right\}$$

重排的规则

重排的规则

规则：最多连续出现 k 篇某种笔记

- 小红书推荐系统的物品分为图文笔记、视频笔记。
- 最多连续出现 $k = 5$ 篇图文笔记，最多连续出现 $k = 5$ 篇视频笔记。
- 如果排 i 到 $i + 4$ 的全部是图文笔记，那么排在 $i + 5$ 的必须是视频笔记。

规则：每 k 篇笔记最多出现 1 篇某种笔记

- 运营推广笔记的精排分会乘以大于 1 的系数 (boost)，帮助笔记获得更多曝光。
- 为了防止 boost 影响体验，限制每 $k = 9$ 篇笔记最多出现 1 篇运营推广笔记。
- 如果排第 i 位的是运营推广笔记，那么排 $i + 1$ 到 $i + 8$ 的不能是运营推广笔记。

规则：前 t 篇笔记最多出现 k 篇某种笔记

- 排名前 t 篇笔记最容易被看到，对用户体验最重要。
(小红书的 top 4 为首屏)
- 小红书推荐系统有带电商卡片的笔记，过多可能会影响体验。
- 前 $t = 1$ 篇笔记最多出现 $k = 0$ 篇带电商卡片的笔记。
- 前 $t = 4$ 篇笔记最多出现 $k = 1$ 篇带电商卡片的笔记。

MMR + 重排规则

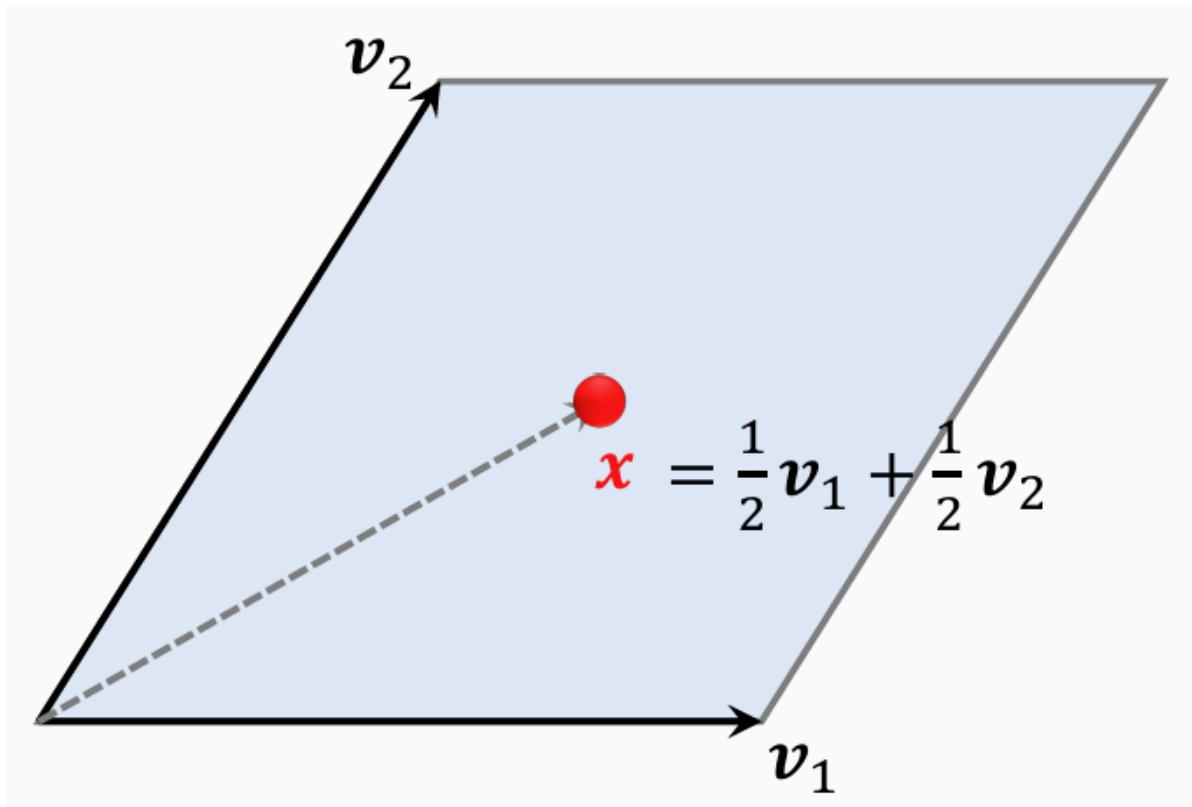
- MMR 每一轮选出一个物品：

$$\arg \max_{i \in \mathcal{R}} \left\{ \theta \cdot \text{reward}_i - (1 - \theta) \cdot \max_{j \in \mathcal{W}} \text{sim}(i, j) \right\}$$

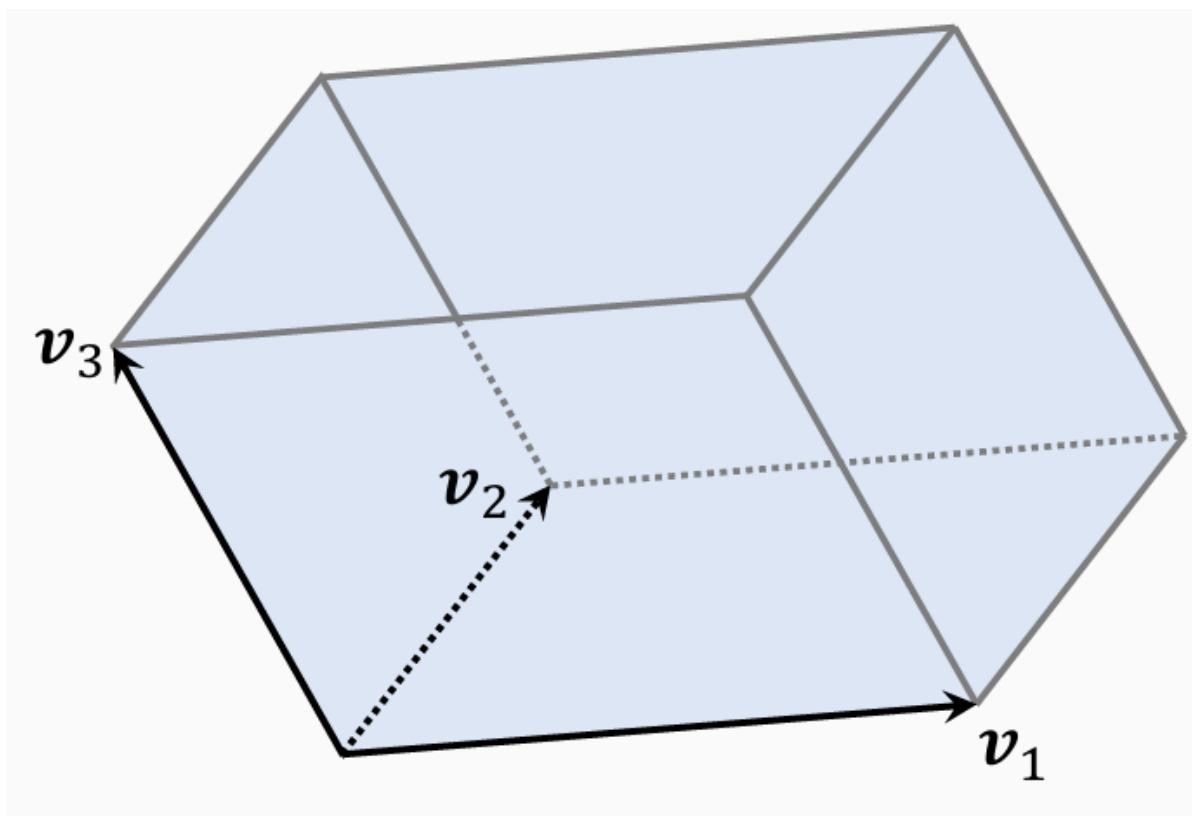
- 重排结合 MMR 与规则，在满足规则的前提下最大化 MRR。
- 每一轮先用规则排除掉 \mathcal{R} 中的部分物品，得到子集 \mathcal{R}' 。
- MMR 公式中的 \mathcal{R} 替换成子集 \mathcal{R}' ，选中的物品符合规则。

DPP：数学基础

超平行体



- 2维空间的超平行体为平行四边形。
 - 平行四边形中的点可以表示为：
- $$\mathbf{x} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2.$$
- 系数 α_1 和 α_2 取值范围是 $[0, 1]$ 。

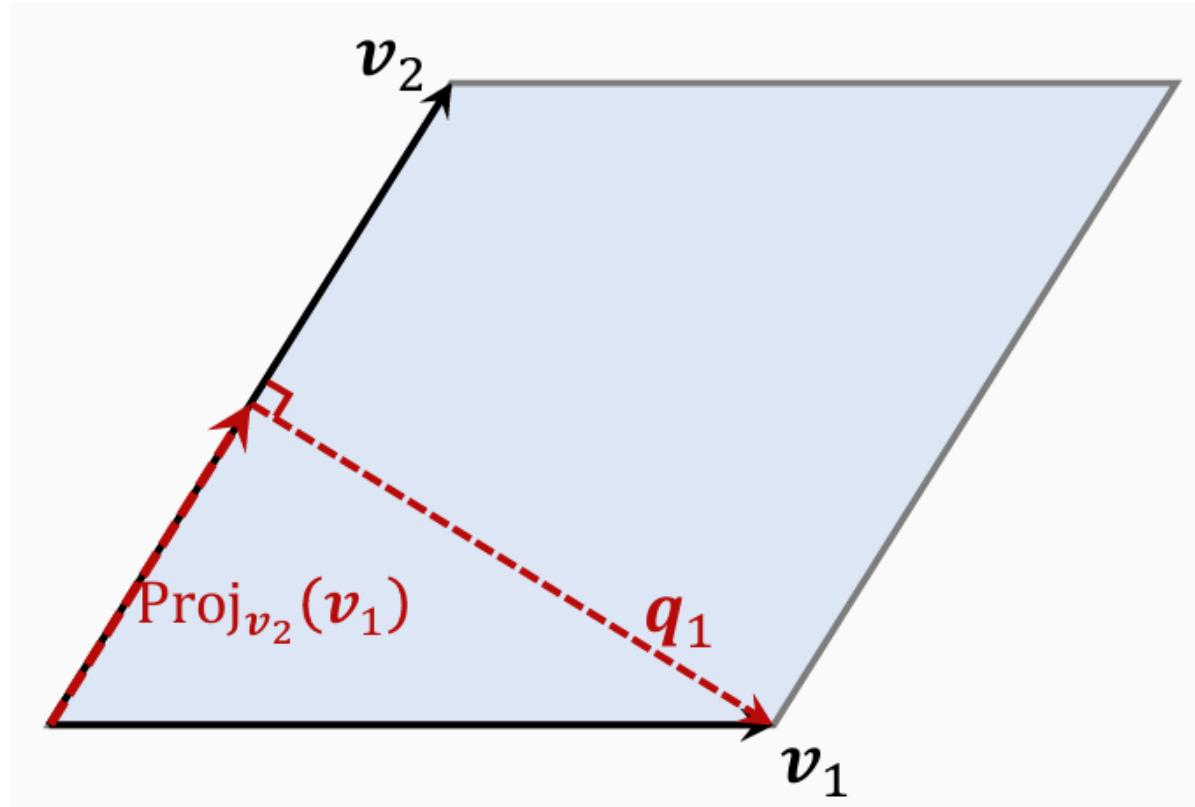


- 3维空间的超平行体为平行六面体。
 - 平行六面体中的点可以表示为：
- $$\mathbf{x} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \alpha_3 \mathbf{v}_3.$$
- 系数 $\alpha_1, \alpha_2, \alpha_3$ 取值范围是 $[0, 1]$ 。

超平行体

- 一组向量 $\mathbf{v}_1, \dots, \mathbf{v}_k \in \mathbb{R}^d$ 可以确定一个 k 维超平行体:
$$P(\mathbf{v}_1, \dots, \mathbf{v}_k) = \{\alpha_1\mathbf{v}_1 + \dots + \alpha_k\mathbf{v}_k \mid 0 \leq \alpha_1, \dots, \alpha_k \leq 1\}.$$
- 要求 $k \leq d$, 比如 $d = 3$ 维空间中有 $k = 2$ 维平行四边形。
- 如果 $\mathbf{v}_1, \dots, \mathbf{v}_k$ 线性相关, 则体积 $\text{vol}(P) = 0$ 。 (例: 有 $k = 3$ 个向量, 落在一个平面上, 则平行六面体的体积为 0。)

平行四边形的面积



以 \mathbf{v}_2 为底, 如何计算高 \mathbf{q}_1 ?

- 计算 \mathbf{v}_1 在 \mathbf{v}_2 上的投影:

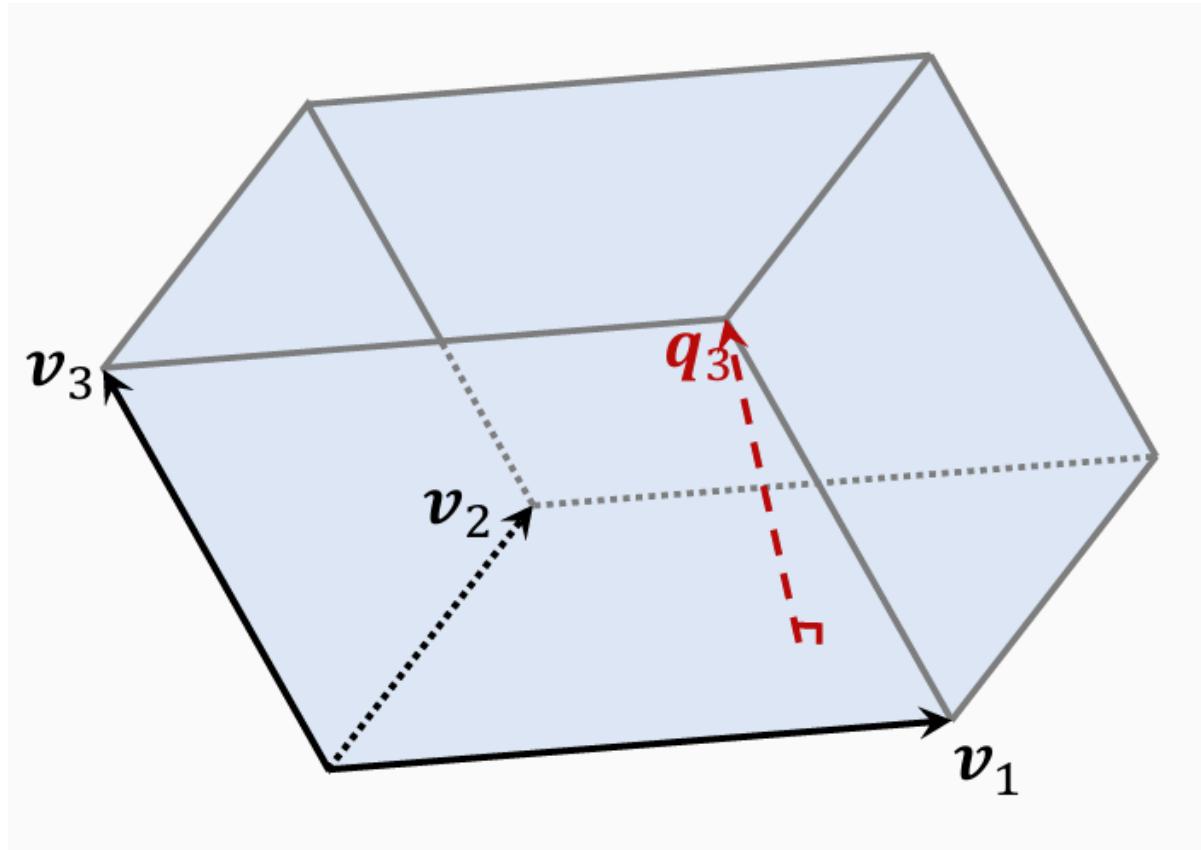
$$\text{Proj}_{\mathbf{v}_2}(\mathbf{v}_1) = \frac{\mathbf{v}_1^T \mathbf{v}_2}{\|\mathbf{v}_2\|_2^2} \cdot \mathbf{v}_2.$$

- 计算

$$\mathbf{q}_1 = \mathbf{v}_1 - \text{Proj}_{\mathbf{v}_2}(\mathbf{v}_1).$$

- 性质: 底 \mathbf{v}_2 与高 \mathbf{q}_1 正交。

平行六面体的体积



- 体积 = 底面积 $\times \|\text{高}\|_2$ 。
- 平行四边形 $P(\mathbf{v}_1, \mathbf{v}_2)$ 是平行六面体 $P(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ 的底。
- 高 \mathbf{q}_3 垂直于底 $P(\mathbf{v}_1, \mathbf{v}_2)$ 。

体积何时最大化、最小化？

- 设 $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ 都是单位向量。
- 当三个向量正交时，平行六面体为正方体，体积最大化， $\text{vol} = 1$ 。
- 当三个向量线性相关时，体积最小化， $\text{vol} = 0$ 。

衡量物品多样性

- 给定 k 个物品，把它们表征为单位向量 $\mathbf{v}_1, \dots, \mathbf{v}_k \in \mathbb{R}^d$ 。($d \geq k$)
- 用超平行体的体积衡量物品的多样性，体积介于 0 和 1 之间。
- 如果 $\mathbf{v}_1, \dots, \mathbf{v}_k$ 两两正交（多样性好），则体积最大化， $\text{vol} = 1$ 。
- 如果 $\mathbf{v}_1, \dots, \mathbf{v}_k$ 线性相关（多样性差），则体积最小化， $\text{vol} = 0$ 。

$$\mathbf{V} = \begin{bmatrix} & & \\ d \times k & & \\ & & \end{bmatrix} \quad \vdots$$

$\mathbf{v}_1 \quad \mathbf{v}_2 \quad \mathbf{v}_k$

- 给定 k 个物品，把它们表征为单位向量 $\mathbf{v}_1, \dots, \mathbf{v}_k \in \mathbb{R}^d$ 。 ($d \geq k$)
- 把它们作为矩阵 $\mathbf{V} \in \mathbb{R}^{d \times k}$ 的列。
- 设 $d \geq k$, 行列式与体积满足:

$$\det(\mathbf{V}^T \mathbf{V}) = \text{vol}(P(\mathbf{v}_1, \dots, \mathbf{v}_k))^2.$$
- 因此，可以用行列式 $\det(\mathbf{V}^T \mathbf{V})$ 衡量向量 $\mathbf{v}_1, \dots, \mathbf{v}_k$ 的多样性。

DPP：多样性算法

多样性问题

- 精排给 n 个物品打分: $\text{reward}_1, \dots, \text{reward}_n$ 。
- n 个物品的向量表征: $\mathbf{v}_1, \dots, \mathbf{v}_n \in \mathbb{R}^d$ 。
- 从 n 个物品中选出 k 个物品，组成集合 \mathcal{S} 。
 - **价值大**: 分数之和 $\sum_{j \in \mathcal{S}} \text{reward}_j$ 越大越好。
 - **多样性好**: \mathcal{S} 中 k 个向量组成的超平行体 $P(\mathcal{S})$ 的体积越大越好。
- 集合 \mathcal{S} 中的 k 个物品的向量作为列，组成矩阵 $\mathbf{V}_{\mathcal{S}} \in \mathbb{R}^{d \times k}$ 。
- 以这 k 个向量作为边，组成超平行体 $P(\mathcal{S})$ 。
- 体积 $\text{vol}(P(\mathcal{S}))$ 可以衡量 \mathcal{S} 中物品的多样性。
- 设 $k \leq d$, 行列式与体积满足:

$$\det(\mathbf{V}_{\mathcal{S}}^T \mathbf{V}_{\mathcal{S}}) = \text{vol}(P(\mathcal{S}))^2$$

行列式点过程 (DPP)

- DPP 是一种传统的统计机器学习方法:

$$\arg \max_{\mathcal{S}: |\mathcal{S}|=k} \log \det(\mathbf{V}_{\mathcal{S}}^T \mathbf{V}_{\mathcal{S}})$$

- Hulu 的论文将 DPP 应用在推荐系统:

$$\arg \max_{\mathcal{S}: |\mathcal{S}|=k} \theta \cdot \left(\sum_{j \in \mathcal{S}} \text{reward}_j \right) + (1 - \theta) \cdot \log \det(\mathbf{V}_{\mathcal{S}}^T \mathbf{V}_{\mathcal{S}})$$

- DPP 应用在推荐系统:

$$\arg \max_{\mathcal{S}:|\mathcal{S}|=k} \theta \cdot \left(\sum_{j \in \mathcal{S}} \text{reward}_j \right) + (1 - \theta) \cdot \log \det(\mathbf{V}_{\mathcal{S}}^T \mathbf{V}_{\mathcal{S}})$$

- 设 \mathbf{A} 为 $n \times n$ 的矩阵, 它的 (i, j) 元素为 $a_{ij} = \mathbf{v}_i^T \mathbf{v}_j$.
- 给定向量 $\mathbf{v}_1, \dots, \mathbf{v}_n \in \mathbb{R}^d$, 需要 $O(n^2 d)$ 时间计算 \mathbf{A} .
- $\mathbf{A}_{\mathcal{S}} = \mathbf{V}_{\mathcal{S}}^T \mathbf{V}_{\mathcal{S}}$ 为 \mathbf{A} 的一个 $k \times k$ 子矩阵。如果 $i, j \in \mathcal{S}$, 则 a_{ij} 是 $\mathbf{A}_{\mathcal{S}}$ 的一个元素。

- DPP 应用在推荐系统:

$$\arg \max_{\mathcal{S}:|\mathcal{S}|=k} \theta \cdot \left(\sum_{j \in \mathcal{S}} \text{reward}_j \right) + (1 - \theta) \cdot \log \det(\mathbf{A}_{\mathcal{S}})$$

- DPP 是个组合优化问题, 从集合 $\{1, \dots, n\}$ 中选出一个大小为 k 的子集 \mathcal{S} .
- 用 \mathcal{S} 表示已选中的物品, 用 \mathcal{R} 表示未选中的物品, 贪心算法求解:

$$\arg \max_{i \in \mathcal{R}} \theta \cdot \text{reward}_i + (1 - \theta) \cdot \log \det(\mathbf{A}_{\mathcal{S} \cup \{i\}})$$

求解DPP

暴力算法

- 贪心算法求解:

$$\arg \max_{i \in \mathcal{R}} \theta \cdot \text{reward}_i + (1 - \theta) \cdot \log \det(\mathbf{A}_{\mathcal{S} \cup \{i\}}).$$

- 计算复杂度分析:

- 对于单个 i , 计算 $\mathbf{A}_{\mathcal{S} \cup \{i\}}$ 的行列式需要 $O(|\mathcal{S}|^3)$ 时间。
- 对于所有的 $i \in \mathcal{R}$, 计算行列式需要时间 $O(|\mathcal{S}|^3 \cdot |\mathcal{R}|)$ 。
- 需要求解上述式子 k 次才能选出 k 个物品。如果暴力计算行列式, 那么总时间复杂度为:

$$O(|\mathcal{S}|^3 \cdot |\mathcal{R}| \cdot k) = O(nk^4).$$

- 暴力算法的总时间复杂度为:

$$O(n^2 d + nk^4).$$

Hulu 的快速算法

- Hulu 的论文设计了一种数值算法, 仅需 $O(n^2 d + nk^2)$ 的时间从 n 个物品中选出 k 个物品。
- 给定向量 $\mathbf{v}_1, \dots, \mathbf{v}_n \in \mathbb{R}^d$, 需要 $O(n^2 d)$ 时间计算 \mathbf{A} .
- 用 $O(nk^2)$ 时间计算所有的行列式 (利用 Cholesky 分解)。
- Cholesky 分解 $\mathbf{A}_{\mathcal{S}} = \mathbf{L} \mathbf{L}^T$, 其中 \mathbf{L} 是下三角矩阵 (对角线以上的元素全零)。
- Cholesky 分解可供计算 $\mathbf{A}_{\mathcal{S}}$ 的行列式:
 - 下三角矩阵 \mathbf{L} 的行列式 $\det(\mathbf{L})$ 等于 \mathbf{L} 对角线元素乘积。
 - $\mathbf{A}_{\mathcal{S}}$ 的行列式为:

$$\det(\mathbf{A}_S) = \det(\mathbf{L})^2 = \prod_i l_{ii}^2.$$

- 已知 $\mathbf{A}_S = \mathbf{L}\mathbf{L}^T$, 则可以快速求出所有 $\mathbf{A}_{S \cup \{i\}}$ 的 Cholesky 分解, 因此可以快速计算出所有 $\mathbf{A}_{S \cup \{i\}}$ 的行列式。
- 贪心算法求解:**

$$\arg \max_{i \in \mathcal{R}} \theta \cdot \text{reward}_i + (1 - \theta) \cdot \log \det(\mathbf{A}_{S \cup \{i\}}).$$

- 初始化:** S 中只有一个物品, \mathbf{A}_S 是 1×1 的矩阵。

- 每一轮循环:**

- 基于上一轮算出的 $\mathbf{A}_S = \mathbf{L}\mathbf{L}^T$, 快速求出 $\mathbf{A}_{S \cup \{i\}}$ 的 Cholesky 分解 ($\forall i \in \mathcal{R}$)。
- 从而求出 $\log \det(\mathbf{A}_{S \cup \{i\}})$ 。

DPP 的扩展

滑动窗口

- 用 S 表示已选中的物品, 用 R 表示未选中的物品, DPP 的贪心算法求解:

$$\arg \max_{i \in \mathcal{R}} \theta \cdot \text{reward}_i + (1 - \theta) \cdot \log \det(\mathbf{A}_{S \cup \{i\}}).$$

- 随着集合 S 增大, 其中相似物品越来越多, 物品向量会趋近线性相关。
- 行列式 $\det(\mathbf{A}_S)$ 会塌缩到零, 对数趋于负无穷。



- 贪心算法:**

$$\arg \max_{i \in \mathcal{R}} \theta \cdot \text{reward}_i + (1 - \theta) \cdot \log \det(\mathbf{A}_{S \cup \{i\}})$$

- 用滑动窗口:**

$$\arg \max_{i \in \mathcal{R}} \theta \cdot \text{reward}_i + (1 - \theta) \cdot \log \det(\mathbf{A}_{W \cup \{i\}})$$

规则约束

- 贪心算法每轮从 R 中选出一个物品:

$$\arg \max_{i \in \mathcal{R}} \theta \cdot \text{reward}_i + (1 - \theta) \cdot \log \det(\mathbf{A}_{W \cup \{i\}})$$

- 有很多规则约束, 例如最多连续出 5 篇视频笔记 (如果已经连续出了 5 篇视频笔记, 下一篇必须是图文笔记)。
- 用规则排除掉 R 中的部分物品, 得到子集 R' , 然后求解:

$$\arg \max_{i \in \mathcal{R}'} \theta \cdot \text{reward}_i + (1 - \theta) \cdot \log \det(\mathbf{A}_{W \cup \{i\}})$$

物品冷启动

物品冷启动：评价指标

物品冷启动

- 小红书上用户新发布的笔记。
- B 站上用户新上传的视频。
- 今日头条上作者新发布的文章。

新笔记冷启动

- 新笔记缺少与用户的交互，导致推荐的难度大、效果差。
- 扶持新发布、低曝光的笔记，可以增强作者发布意愿。

优化冷启的目标

1. 精准推荐：克服冷启的困难，把新笔记推荐给合适的用户，不引起用户反感。
2. 激励发布：流量向低曝光新笔记倾斜，激励作者发布。
3. 挖掘高潜：通过初期小流量的试探，找到高质量的笔记，给予流量倾斜。

评价指标

- 作者侧指标：
 - 发布渗透率、人均发布量。
- 用户侧指标：
 - 新笔记指标：新笔记的点击率、交互率。
 - 大盘指标：消费时长、日活、月活。
- 内容侧指标：
 - 高热笔记占比。

作者侧指标

发布渗透率 (penetration rate)

- 发布渗透率 = 当日发布人数 / 日活人数
- 发布一篇或以上，就算一个发布人数。
- 例：
 - 当日发布人数 = 100 万
 - 日活人数 = 2000 万
 - 发布渗透率 = $100 / 2000 = 5\%$

人均发布量

- 人均发布量 = 当日发布笔记数 / 日活人数
- 例：
 - 每日发布笔记数 = 200 万
 - 日活人数 = 2000 万
 - 人均发布量 = $200 / 2000 = 0.1$

发布渗透率、人均发布量反映出作者的发布积极性。

冷启的重要优化目标是促进发布，增大内容池。

新笔记获得的曝光越多，首次曝光和交互出现得越早，作者发布积极性越高。

用户侧指标

新笔记的消费指标

- 新笔记的点击率、交互率。
 - 问题：曝光的基尼系数很大。
 - 少数头部新笔记占据了大部分的曝光。
- 分别考察高曝光、低曝光新笔记。
 - 高曝光：比如 >1000 次曝光。
 - 低曝光：比如 <1000 次曝光。

内容侧指标

高热笔记占比

- 高热笔记：前 30 天获得 1000+ 次点击。
- 高热笔记占比越高，说明冷启阶段挖掘优质笔记的能力越强。

总结

- **作者侧指标**：发布渗透率、人均发布量。
- **用户侧指标**：新笔记消费指标、大盘消费指标。
- **内容侧指标**：高热笔记占比。

冷启动的优化点

- **优化全链路**（包括召回和排序）。
- **流量调控**（流量怎么在新物品、老物品中分配）。

物品冷启动：简单的召回通道

召回的依据

冷启召回的困难

- 缺少用户交互，还没学好笔记 ID embedding，导致双塔模型效果不好。
- 缺少用户交互，导致 ItemCF 不适用。

双塔模型

ID Embedding

改进方案 1：新笔记使用 default embedding

- 物品塔做 ID embedding 时，让所有新笔记共享一个 ID，而不是用自己真正的 ID。
- Default embedding：共享的 ID 对应的 embedding 向量。
- 到下次模型训练的时候，新笔记才有自己的 ID embedding 向量。

改进方案 2：利用相似笔记 embedding 向量

- 查找 top k 内容最相似的高曝光笔记。
- 把 k 个高曝光笔记的 embedding 向量取平均，作为新笔记的 embedding。

多个向量召回池

- 多个召回池，让新笔记有更多曝光机会。
 - 1 小时新笔记，
 - 6 小时新笔记，
 - 24 小时新笔记，
 - 30 天笔记。
- 共享同一个双塔模型，那么多个召回池不增加训练的代价。

类目召回

基于类目的召回

- 系统维护类目索引：
类目 → 笔记列表（按时间倒排）
- 用类目索引做召回：
用户画像 → 类目 → 笔记列表
- 取回笔记列表上前 k 篇笔记（即最新的 k 篇）。

基于关键词的召回

- 系统维护关键词索引：
关键词 → 笔记列表（按时间倒排）
- 根据用户画像上的 \text{\color{red}{关键词}} 做召回。

缺点

- 缺点 1：只对刚刚发布的新笔记有效。
 - 取回某类目 / 关键词下最新的 k 篇笔记。
 - 发布几小时之后，就再没有机会被召回。
- 缺点 2：弱个性化，不够精准。

物品冷启动：聚类召回

聚类召回

基本思想

- 如果用户喜欢一篇笔记，那么他会喜欢内容相似的笔记。
- 事先训练一个神经网络，基于笔记的类别和图文内容，把笔记映射到向量。
- 对笔记向量做聚类，划分为 1000 个 *cluster*，记录每个 *cluster* 的中心方向。（*k-means* 聚类，用余弦相似度。）

聚类索引

- 一篇新笔记发布之后，用神经网络把它映射到一个特征向量。
- 从 1000 个向量（对应 1000 个 *cluster*）中找到最相似的向量，作为新笔记的 *cluster*。

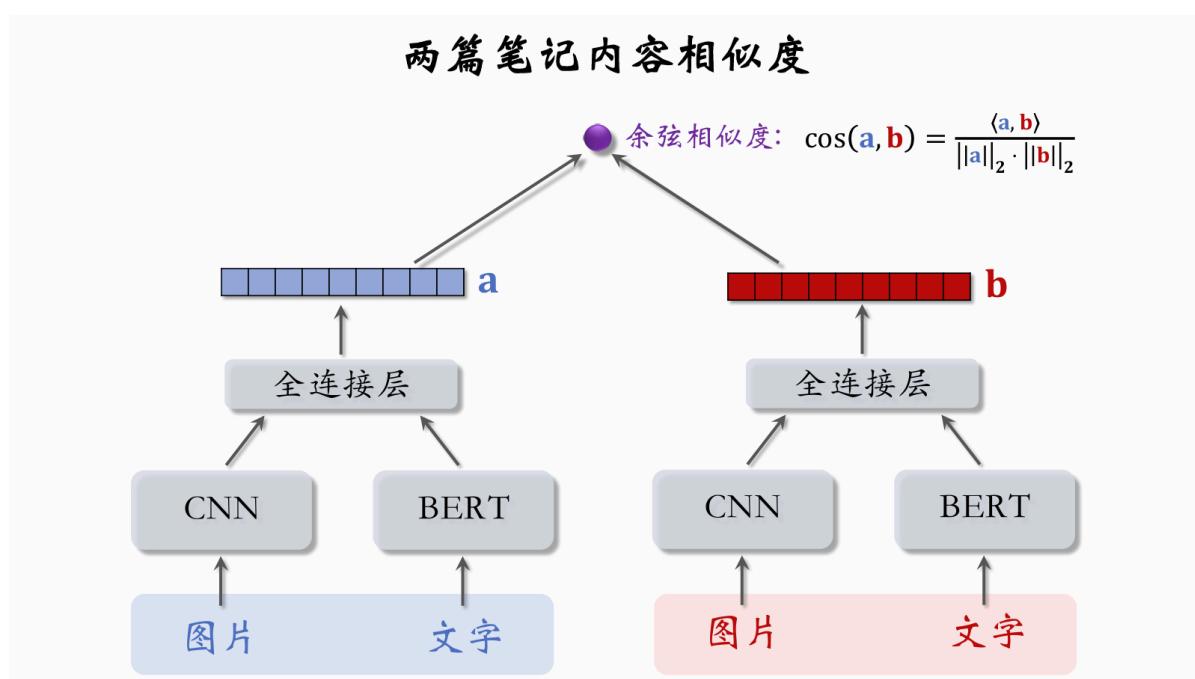
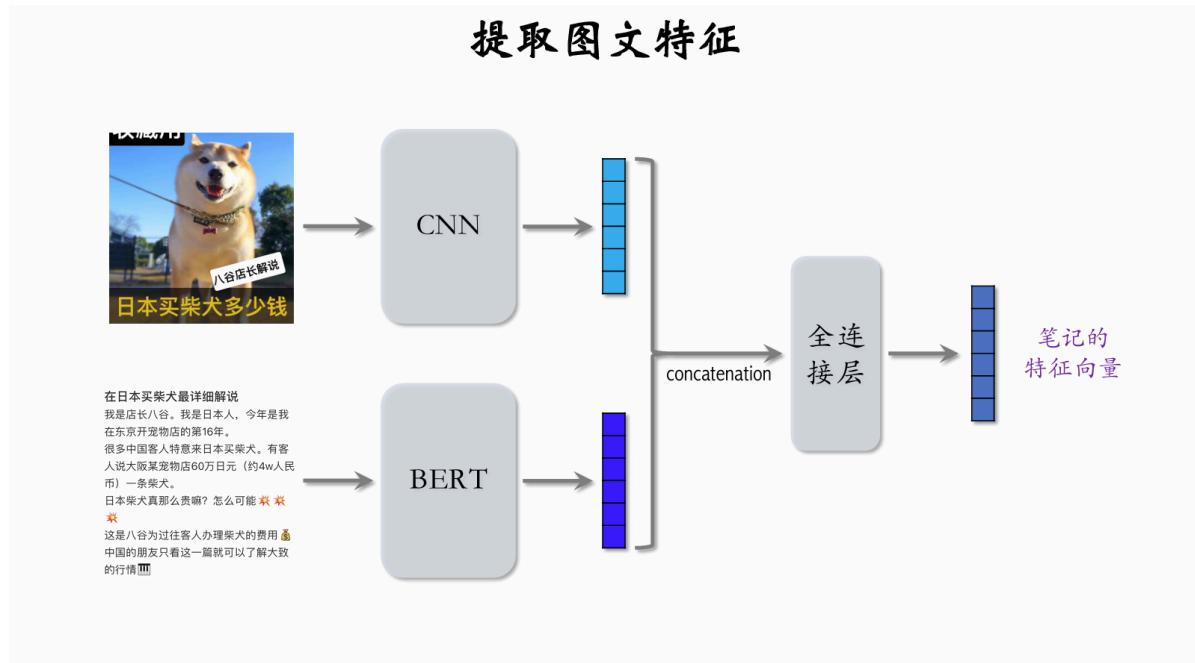
- 索引:

cluster → 笔记ID列表（按时间倒排）

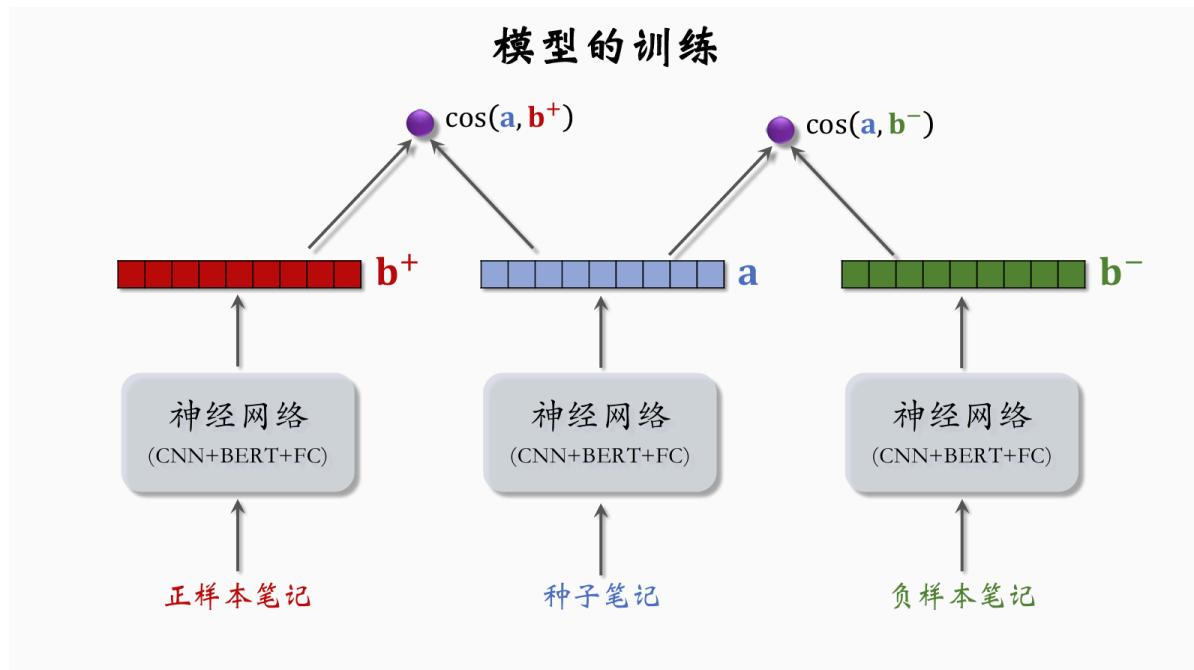
线上召回

- 给定用户 ID，找到他的 last- n 交互的笔记列表，把这些笔记作为种子笔记。
- 把每篇种子笔记映射到向量，寻找最相似的 *cluster*。（知道了用户对哪些 *cluster* 感兴趣。）
- 从每个 *cluster* 的笔记列表中，取回最新的 m 篇笔记。
- 最多取回 mn 篇新笔记。

内容相似度模型



训练内容相似度模型



模型的训练

基本想法：鼓励 $\cos(\mathbf{a}, \mathbf{b}^+)$ 大于 $\cos(\mathbf{a}, \mathbf{b}^-)$

Triplet hinge loss:

$$L(\mathbf{a}, \mathbf{b}^+, \mathbf{b}^-) = \max\{0, \cos(\mathbf{a}, \mathbf{b}^-) + m - \cos(\mathbf{a}, \mathbf{b}^+)\}$$

Triplet logistic loss:

$$L(\mathbf{a}, \mathbf{b}^+, \mathbf{b}^-) = \log(1 + \exp(\cos(\mathbf{a}, \mathbf{b}^-) - \cos(\mathbf{a}, \mathbf{b}^+)))$$

<种子笔记, 正样本>

方法一：人工标注二元组的相似度

方法二：算法自动选正样本

筛选条件：

- 只用高曝光笔记作为二元组（因为有充足的用户交互信息）。
- 两篇笔记有相同的二级类别，比如都是“菜谱教程”。
- 用 ItemCF 的物品相似度选正样本。

<种子笔记, 负样本>

- 从全体笔记中随机选出满足条件的：
 - 字数较多（神经网络提取的文本信息有效）。
 - 笔记质量高，避免图文无关。

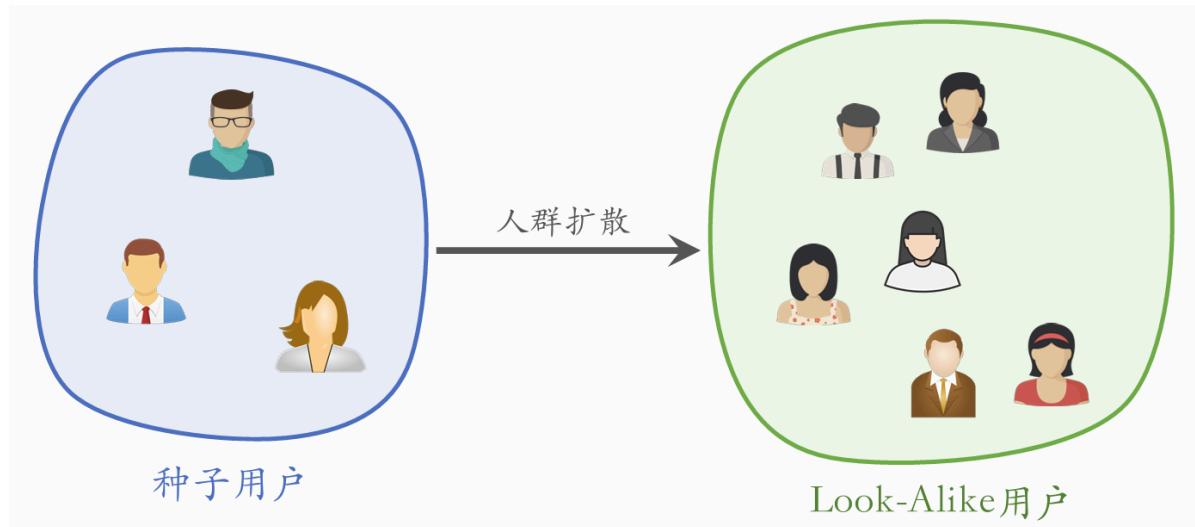
总结

- **基本思想**：根据用户的点赞、收藏、转发记录，推荐内容相似的笔记。
- **线下训练**：多模态神经网络把图文内容映射到向量。
- **线上服务**：

用户喜欢的笔记 → 特征向量 → 最近的 Cluster → 新笔记

物品冷启动：Look-Alike 人群扩散

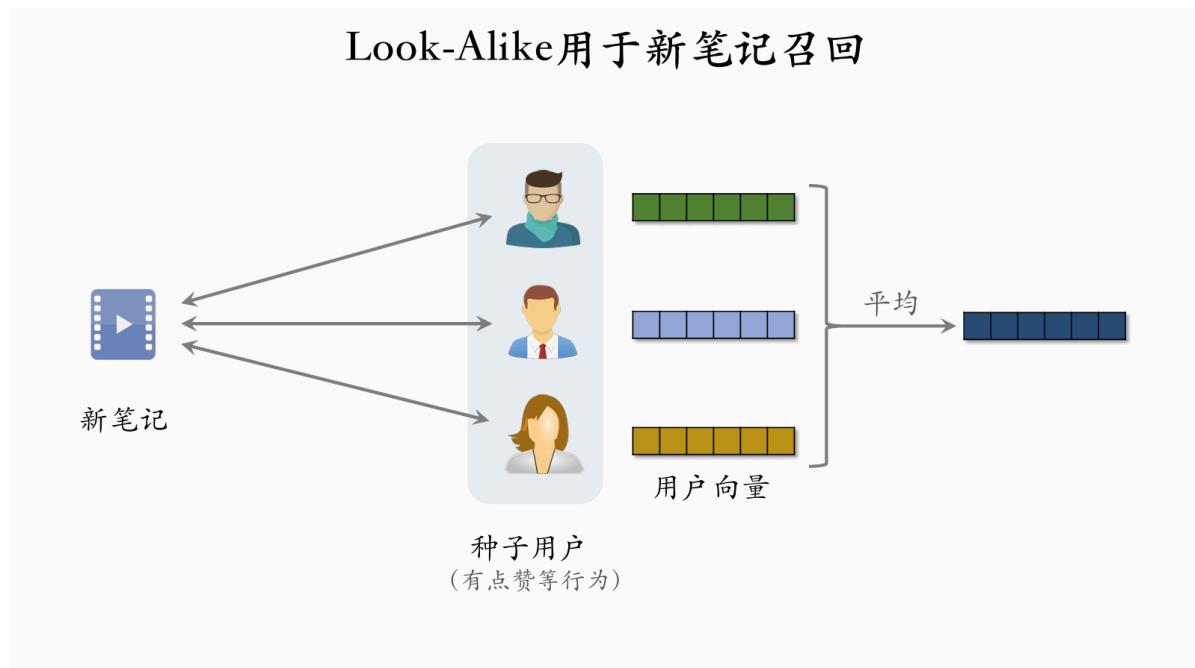
Look-Alike 起源于互联网广告



- 如何计算两个用户的相似度？
- UserCF：两个用户有共同的兴趣点。
- Embedding：两个用户向量的 cos 较大。

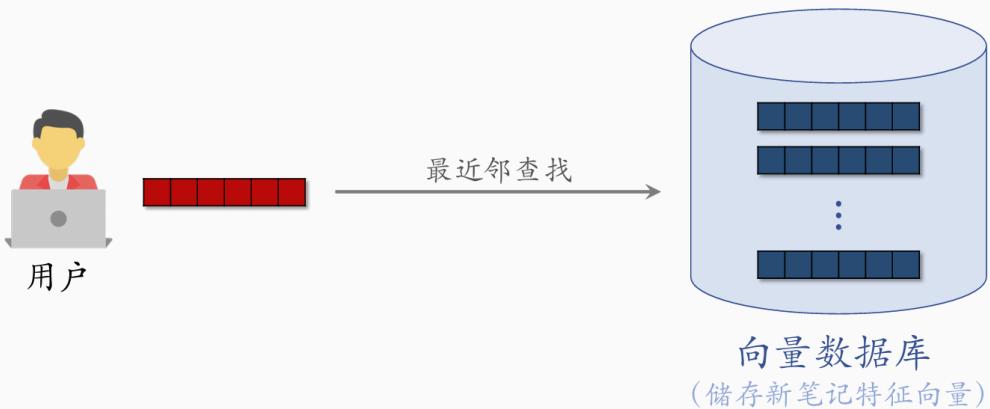
Look-Alike 用于新笔记召回

- 点击、点赞、收藏、转发 —— 用户对笔记可能感兴趣。
- 把有交互的用户作为新笔记的种子用户。
- 用 look-alike 在相似用户中扩散。



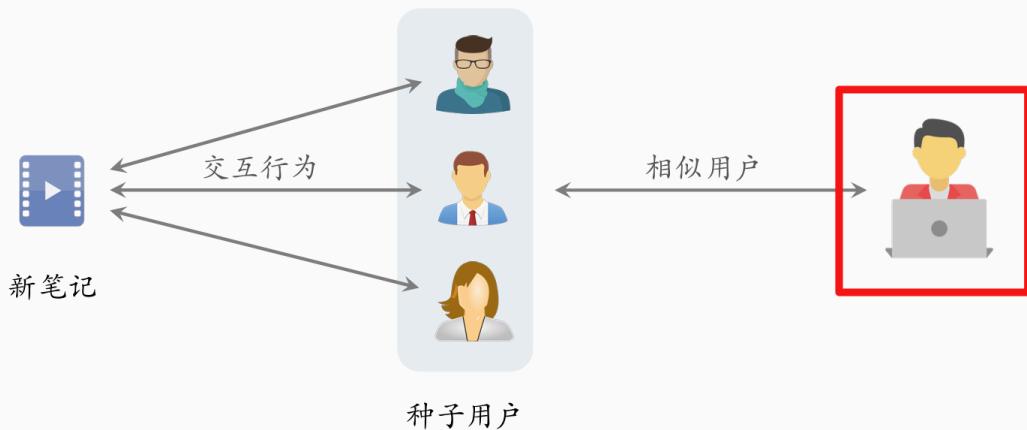
- 近线更新特征向量。
- 特征向量是有交互的用户的向量的平均。
- 每当有用户交互该物品，更新笔记的特征向量。

Look-Alike用于新笔记召回



利用双塔模型计算出用户的特征向量，将这个特征向量在向量数据库中做最近邻查找。这个过程就叫做 Look-Alike 召回。

Look-Alike用于新笔记召回



如果种子用户喜欢某篇笔记，那么相似用户也可能喜欢这篇笔记，这叫做 Look-Alike 扩散召回通道。

物品冷启动：流量调控

扶持新笔记的目的

- **目的1：**促进发布，增大内容池。
 - 新笔记获得的曝光越多，作者创作积极性越高。
 - 反映在发布渗透率、人均发布量。
- **目的2：**挖掘优质笔记。
 - 做探索，让每篇新笔记都能获得足够曝光。
 - 挖掘的能力反映在高热笔记占比。

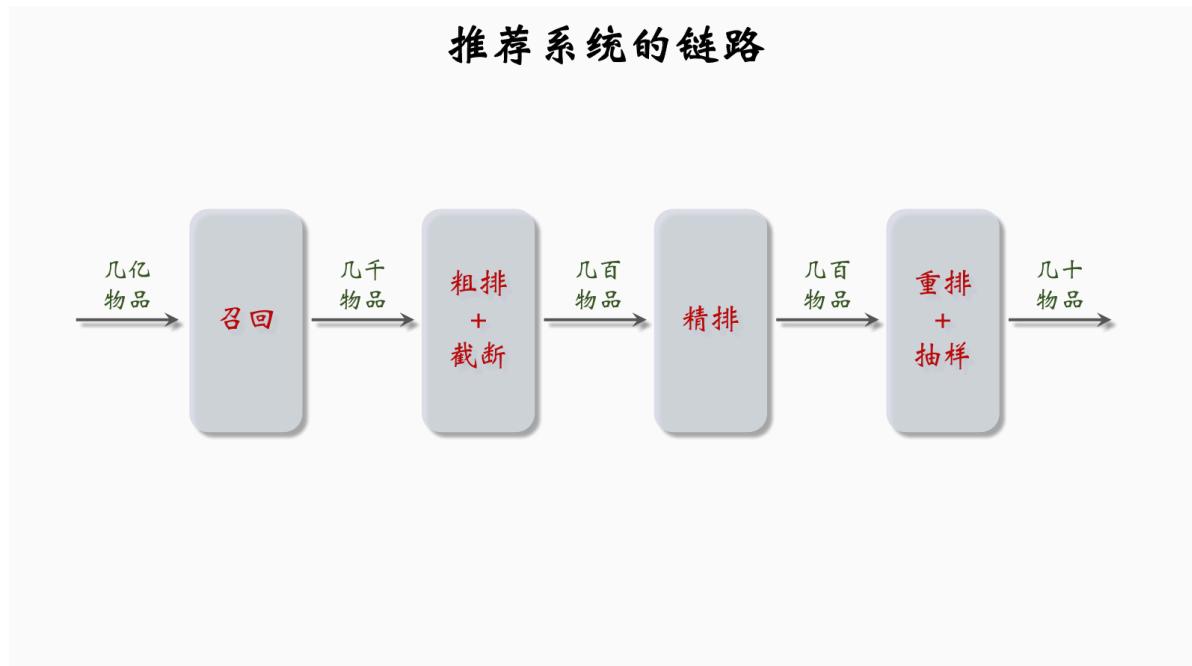
工业界的做法

- 假设推荐系统只分发年龄 <30 天的笔记。
- 假设采用自然分发，新笔记（年龄 <24 小时）的曝光占比为 1/30。
- 扶持新笔记，让新笔记的曝光占比远大于 1/30。

流量调控技术的发展

1. 在推荐结果中强插新笔记。
2. 对新笔记的排序分数做提权（boost）。
3. 通过提权，对新笔记做保量。
4. 差异化保量。

新笔记提权（boost）



新笔记提权

- **目标：**让新笔记有更多机会曝光。
 - 如果做自然分发，24 小时新笔记占比为 1/30。
 - 做人为干涉，让新笔记占比大幅提升。
- 干涉粗排、重排环节，给新笔记提权。
- **优点：**容易实现，投入产出比好。
- **缺点：**
 - 曝光量对提权系数很敏感。
 - 很难精确控制曝光量，容易过度曝光和不充分曝光。

新笔记保量

- **保量：**不论笔记质量高低，都保证 24 小时获得 100 次曝光。
- 在原有提权系数的基础上，乘以额外的提权的系数，例如：

发布时间	当前曝光次数			
	0~24次	25~49次	50~74次	75~100次
0~5小时	1.0	1.0	1.0	1.0
6~11小时	1.1	1.0	1.0	1.0
12~17小时	1.2	1.1	1.0	1.0
18~24小时	1.3	1.2	1.1	1.0

动态提权保量

用下面四个值计算提权系数

- 目标时间：比如 24 小时。
- 目标曝光：比如 100 次。
- 发布时间：比如笔记已经发布 12 小时。
- 已有曝光：比如笔记已经获得 20 次曝光。

计算公式：

$$\text{提权系数} = f\left(\frac{\text{发布时间}}{\text{目标时间}}, \frac{\text{已有曝光}}{\text{目标曝光}}\right) = f(0.5, 0.2)$$

保量的难点

保量成功率远低于 100%

- 很多笔记在 24 小时达不到 100 次曝光。
- 召回、排序存在不足。
- 提权系数调得不好。

线上环境变化会导致保量失败

- 线上环境变化：新增召回通道、升级排序模型、改变重排打散规则……
- 应对措施：线上环境变化后，需要调整提权系数。

给新笔记分数 boost 越多，对新笔记越有利？

- 好处：分数提升越多，曝光次数越多。
- 坏处：把笔记推荐给不太合适的受众。
 - 提权系数过高，导致预估兴趣分数偏高，会将笔记推荐给不合适的受众
 - 点击率、点赞率等指标会偏低。
 - 长期会受推荐系统打压，难以成长为热门笔记。

差异化保量

- **保量**：不论新笔记质量高低，都做扶持，在前 24 小时给 100 次曝光。
- **差异化保量**：不同笔记有不同保量目标，普通笔记保 100 次曝光，内容优质的笔记保 100~500 次曝光。

差异化保量

- **基础保量**: 24 小时 100 次曝光。
- **内容质量**: 用模型评价内容质量高低，给予额外保量目标，上限是加 200 次曝光。
- **作者质量**: 根据作者历史上的笔记质量，给予额外保量目标，上限是加 200 次曝光。
- **一篇笔记最少有 100 次保量，最多有 500 次保量。**

总结

- **流量调控**: 流量怎么在新老笔记之间分配。
- **扶持新笔记**: 单独的召回通道，在排序阶段提权。
- **保量**: 帮助新笔记在前 24 小时获得 100 次曝光。
- **差异化保量**: 根据内容质量、作者质量，决定保量目标。

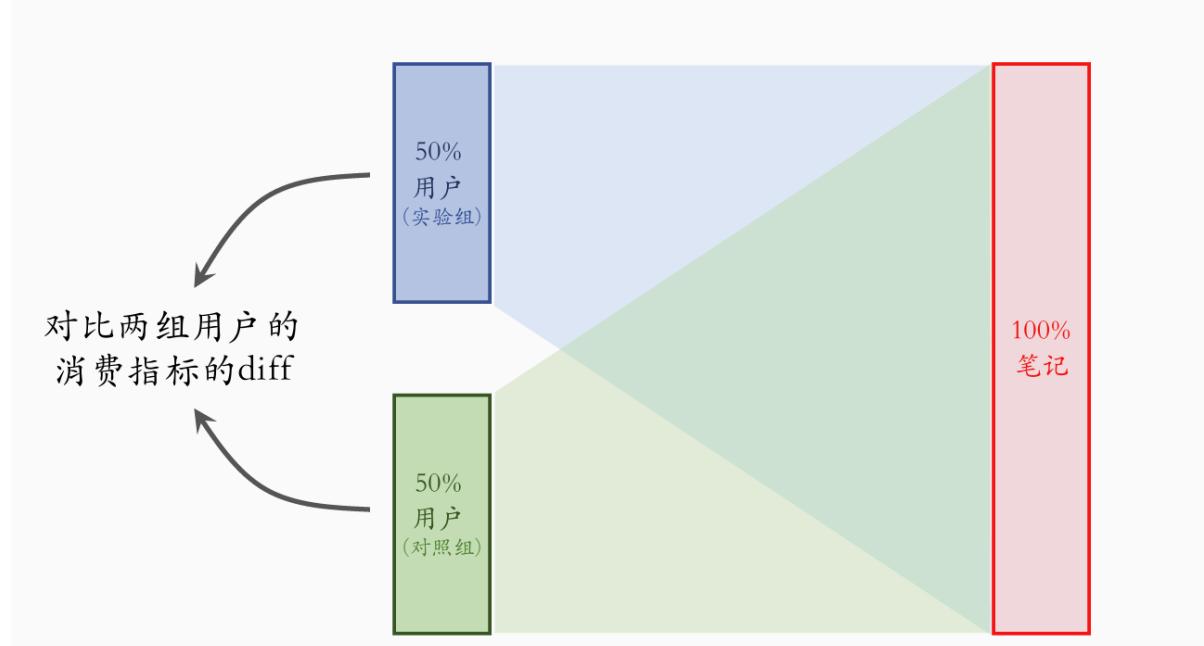
物品冷启动：AB测试

新笔记冷启的 AB 测试

- **作者侧指标**:
 - 发布渗透率、人均发布量。
- **用户侧指标**:
 - 对新笔记的点击率、交互率。
 - 大盘指标：消费时长、日活、月活。

用户侧实验

推荐系统标准的AB测试

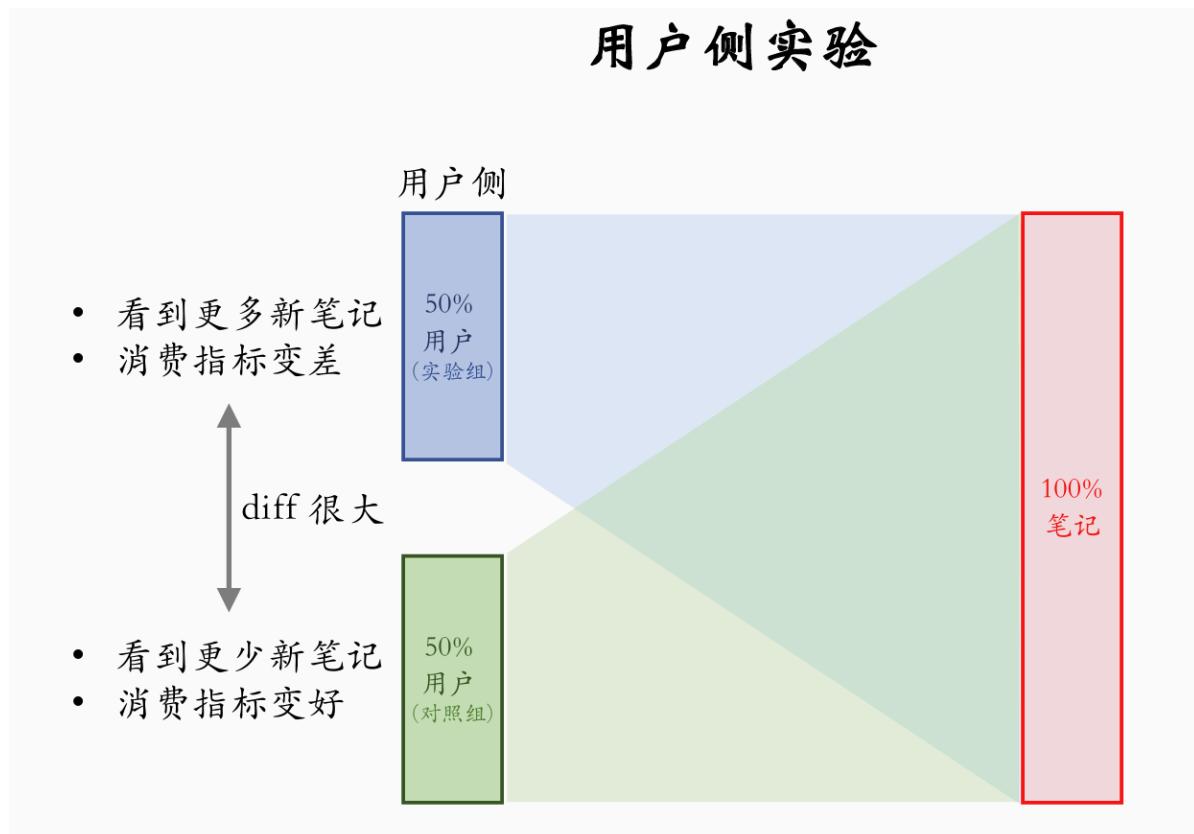


用户侧实验

缺点

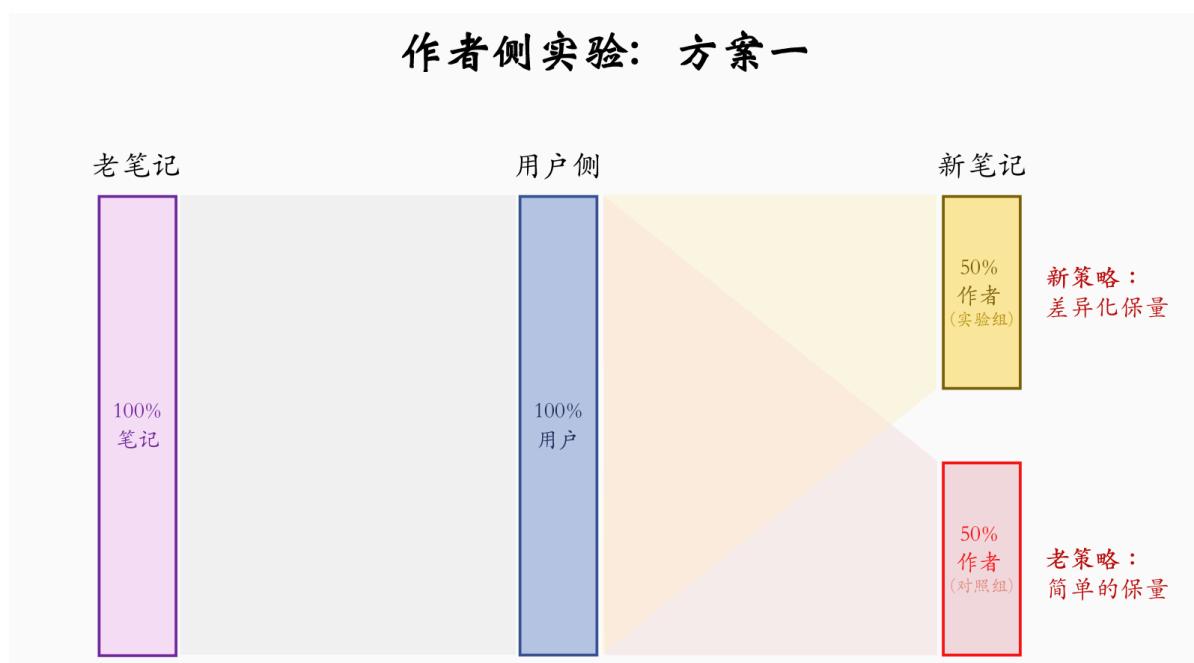
- **限定**: 保量 100 次曝光。
- **假设**: 新笔记曝光越多，用户使用 APP 时长越低。

- 新策略：把新笔记排序时的权重增大两倍。
- 结果（只看消费指标）
 - AB 测试的 diff 是负数（策略组不如对照组）。
 - 如果推全，diff 会缩小（比如 $-2\% \rightarrow -1\%$ ）。
 - 这是因为新笔记采取保量。实验组的新笔记曝光量偏多，对照组偏少。比如有 90 篇新笔记，每篇保量 100 次，一共曝光 9000 次。实验组曝光了 6000 次，对照组曝光了 3000 次。试验结束后，原来实验组的 50% 用户曝光了 4500 次，对照组的 50% 用户也曝光了 4500 次，因此实验计算的 diff 偏大。



作者侧实验

作者侧实验：方案一



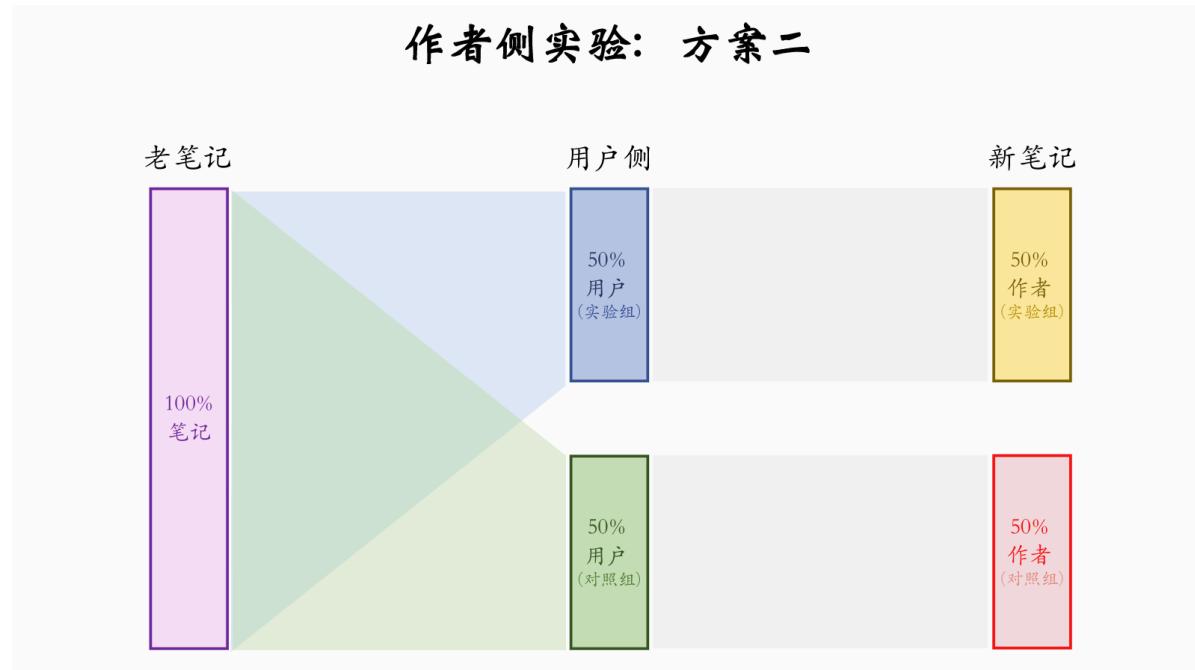
缺点：新笔记之间会抢流量

- 设定：
 - 新老笔记走各自队列，没有竞争。
 - 重排：分给新笔记 1/3 流量，分给老笔记 2/3 流量。
- 新策略：把新笔记的权重增大两倍。
- 结果（只看发布指标）：
 - AB 测试的 diff 是正数（策略组优于对照组）。
 - 如果推全，diff 会消失（比如 2% → 0）。

缺点：新笔记和老笔记抢流量

- 设定：新老笔记自由竞争。
- 新策略：把新笔记排序时的权重增大两倍。
- AB 测试时，50% 新笔记（带策略）跟 100% 老笔记抢流量。
- 推全后，100% 新笔记（带策略）跟 100% 老笔记抢流量。
- 作者侧 AB 测试结果与推全结果有些差异。

作者侧实验：方案二

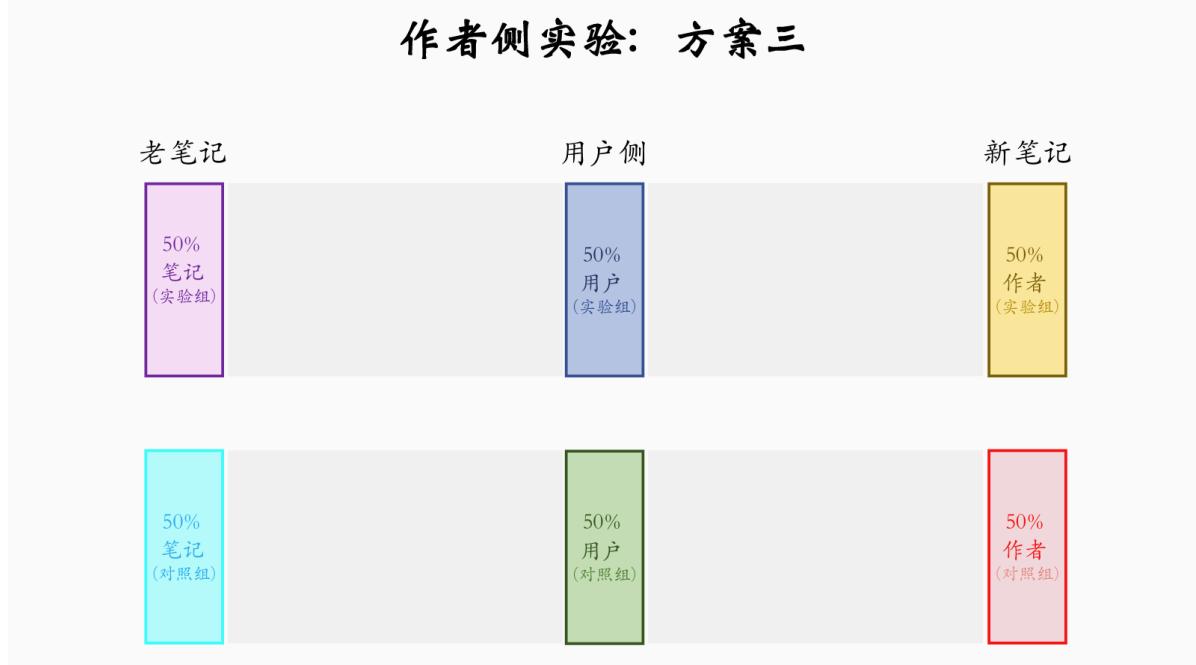


方案二比方案一的优缺点

- 优点：新笔记的两个桶不抢流量，作者侧实验结果更可信。
- 相同：新笔记和老笔记抢流量，作者侧 AB 测试结果与推全结果有些差异。
- 缺点：新笔记池减小一半，对用户体验造成负面影响。

作者侧实验：方案三

作者侧实验：方案三



对公司业务造成影响。

总结

- 冷启的 AB 测试需要观测 作者发布指标 和 用户消费指标。
- 各种 AB 测试的方案都有缺陷。 (小红书有更好的方案，但也不完美。)
- 设计方案的时候，问自己几个问题：
 - 实验组、对照组新笔记会不会抢流量？
 - 新笔记、老笔记怎么抢流量？
 - 同时隔离笔记、用户，会不会让内容池变小？
 - 如果对新笔记做保量，会发生什么？

涨指标的方法

涨指标的方法

推荐系统的评价指标

- 日活用户数 (DAU) 和留存是最核心的指标。
- 目前工业界最常用 $LT7$ 和 $LT30$ 衡量留存。
 - 某用户今天 (t_0) 登录 APP，未来 7 天 ($t_0 \sim t_6$) 中有 4 天登录 APP，那么该用户今天 (t_0) 的 $LT7$ 等于 4。
 - 显然有 $1 \leq LT7 \leq 7$ 和 $1 \leq LT30 \leq 30$ 。
 - LT 增长通常意味着用户体验提升。 (除非 LT 增长且 DAU 下降。)
 - 假设 APP 禁止低活用户登录，则 DAU 下降， LT 增长。
- 其他核心指标：用户使用时长、总阅读数（即总点击数）、总曝光数。这些指标的重要性低于 DAU 和留存。
 - 时长增长， LT 通常会增长。
 - 时长增长，阅读数、曝光数可能会下降。
- 非核心指标：点击率、交互率、等等。

- 对于 UGC 平台，发布量和发布渗透率也是核心指标。

涨指标的方法有哪些？

1. 改进召回模型，添加新的召回模型。
2. 改进粗排和精排模型。
3. 提升召回、粗排、精排中的多样性。
4. 特殊对待新用户、低活用户等特殊人群。
5. 利用关注、转发、评论这三种交互行为。

涨指标的方法：召回

召回模型 & 召回通道

- 推荐系统有几十条召回通道，它们的召回总量是固定的。总量越大，指标越好，粗排计算量越大。
- 双塔模型 (*two-tower*) 和 *item-to-item* (*I2I*) 是最重要的两类召回模型，占据召回的大部分配额。
- 有很多小众的模型，占据的配额很少。在召回总量不变的前提下，添加某些召回模型可以提升核心指标。
- 有很多内容池，比如 30 天物品、1 天物品、6 小时物品、新用户优质内容池、分人群内容池。
- 同一个模型可以用于多个内容池，得到多条召回通道。

改进双塔模型

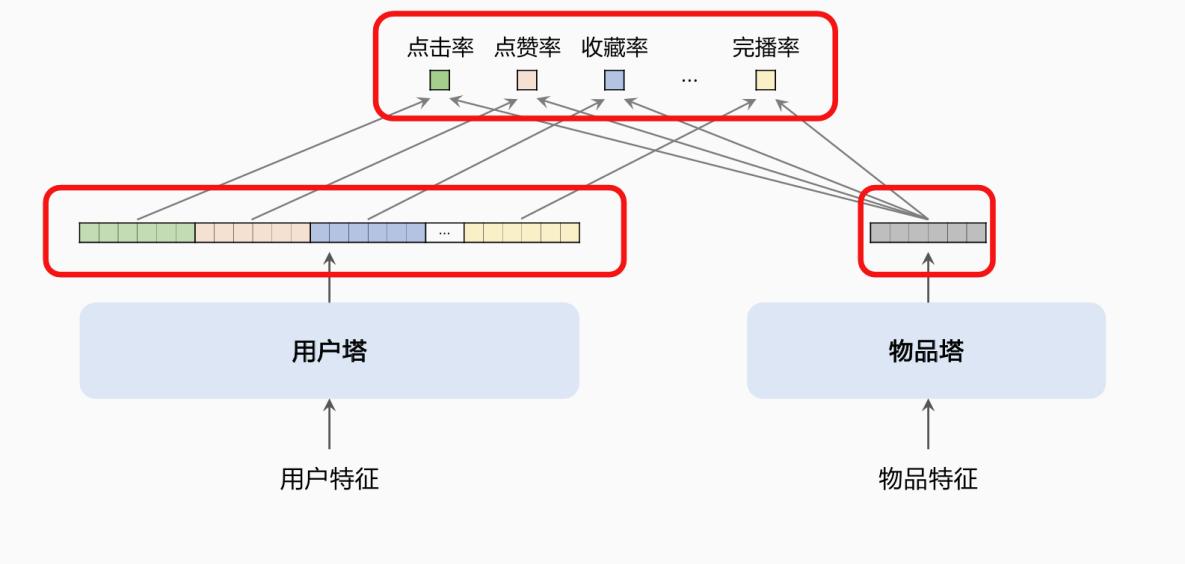
方向1：优化正样本、负样本。

- **简单正样本**：有点击的（用户，物品）二元组。
- **简单负样本**：随机组合的（用户，物品）二元组。
- **困难负样本**：排序靠后的（用户，物品）二元组。

方向2：改进神经网络结构。

- **Baseline**：用户塔、物品塔分别是全连接网络，各输出一个向量，分别作为用户、物品的表征。
- **改进**：用户塔、物品塔分别用 *DCN* 代替全连接网络。
- **改进**：在用户塔中使用用户行为序列 (*last-n*)。
- **改进**：使用多向量模型代替单向量模型。（标准的双塔模型也叫单向量模型。）

改进双塔模型



方向3：改进模型的训练方法。

- **Baseline**: 做二分类，让模型学会区分正样本和负样本。
- **改进**: 结合二分类、batch 内负采样。 (对于 batch 内负采样，需要做纠偏。)
- **改进**: 使用自监督学习方法，让冷门物品的 embedding 学得更好。

Item-to-Item (I2I)

- *I2I* 是一大类模型，基于相似物品做召回。
- 最常见的用法是 *U2I2I* ($user \rightarrow item \rightarrow item$)。
 - 用户 u 喜欢物品 i_1 (用户历史上交互过的物品)。
 - 寻找 i_1 的相似物品 i_2 (即 *I2I*)。
 - 将 i_2 推荐给 u 。
- 如何计算物品相似度？
- 方法1: ItemCF 及其变体。
 - 一些用户同时喜欢物品 i_1 和 i_2 ，则认为 i_1 和 i_2 相似。
 - *ItemCF*、*Online ItemCF*、*Swing*、*Online Swing* 都是基于相同的思想。
 - 线上同时使用上述 4 种 *I2I* 模型，各分配一定配额。
- 方法2: 基于物品向量表征，计算向量相似度。（双塔模型、图神经网络均可计算物品向量表征。）

小众的召回模型

类似 I2I 的模型

- **U2U2I** ($user \rightarrow user \rightarrow item$) : 已知用户 u_1 与 u_2 相似，且 u_2 喜欢物品 i ，那么给用户 u_1 推荐物品 i 。
- **U2A2I** ($user \rightarrow author \rightarrow item$) : 已知用户 u 喜欢作者 a ，且 a 发布物品 i ，那么给用户 u 推荐物品 i 。
- **U2A2A2I** ($user \rightarrow author \rightarrow author \rightarrow item$) : 已知用户 u 喜欢作者 a_1 ，且 a_1 与 a_2 相似， a_2 发布物品 i ，那么给用户 u 推荐物品 i 。

总结：改进召回模型

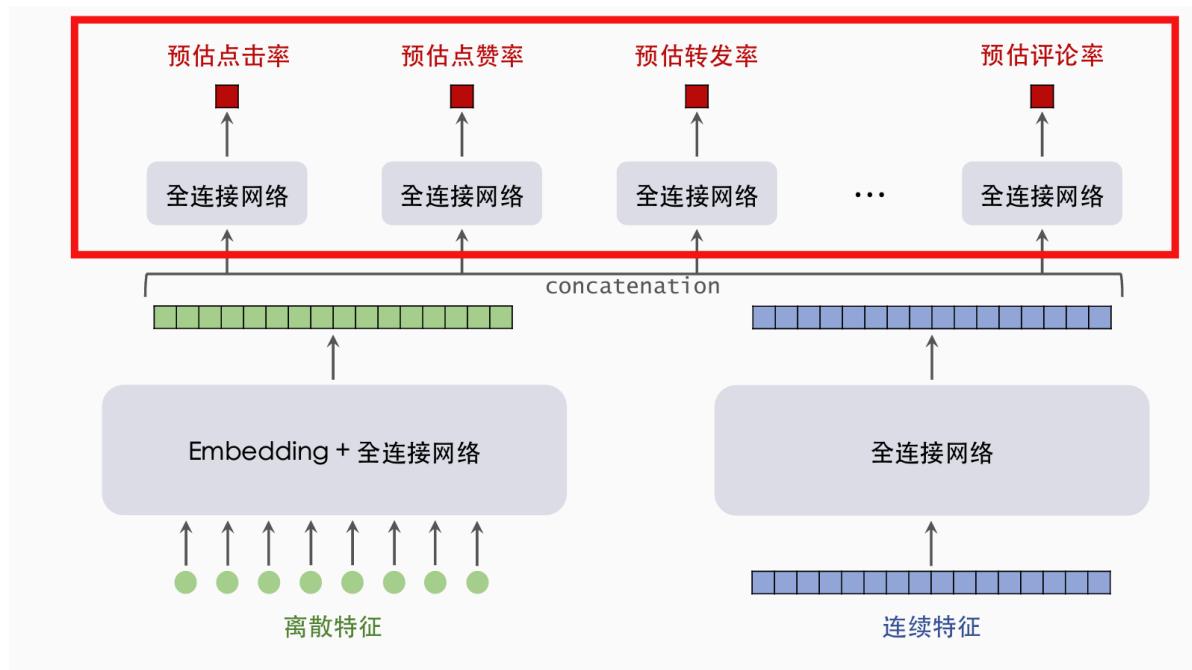
- **双塔模型**：优化正负样本、改进神经网络结构、改进训练的方法。
- **I2I 模型**：同时使用 *ItemCF* 及其变体，使用物品向量表征计算物品相似度。
- **添加小众的召回模型**，比如 *PDN*、*Deep Retrieval*、*SINE*、*M2GRL* 等模型。
- 在召回总量不变的前提下，调整各召回通道的配额。（可以让各用户群体用不同的配额。）

涨指标的方法：排序模型

排序模型

1. 精排模型的改进
2. 粗排模型的改进
3. 用户行为序列建模
4. 在线学习
5. 老汤模型

精排模型的改进



精排模型：基座

- 基座的输入包括离散特征和连续特征，输出一个向量，作为多目标预估的输入。
- **改进 1：**基座加宽加深，计算量更大，预测更准确。
- **改进 2：**做自动的特征交叉，比如 *bilinear* [1] 和 *LHUC* [2]。
- **改进 3：**特征工程，比如添加统计特征、多模态内容特征。

精排模型：多目标预估

- 基于基座输出的向量，同时预估点击率等多个目标。
- **改进 1：**增加新的预估目标，并把预估结果加入融合公式。
 - 最标准的目标包括点击率、点赞率、收藏率、转发率、评论率、关注率、完播率……

- 寻找更多目标，比如进入评论区、给他人写的评论点赞……
- 把新的预估目标加入融合公式。
- **改进 2：***MMoE*、*PLE* 等结构可能有效，但往往无效。
- **改进 3：**纠正 *position bias* 可能有效，也可能无效。

粗排模型的改进

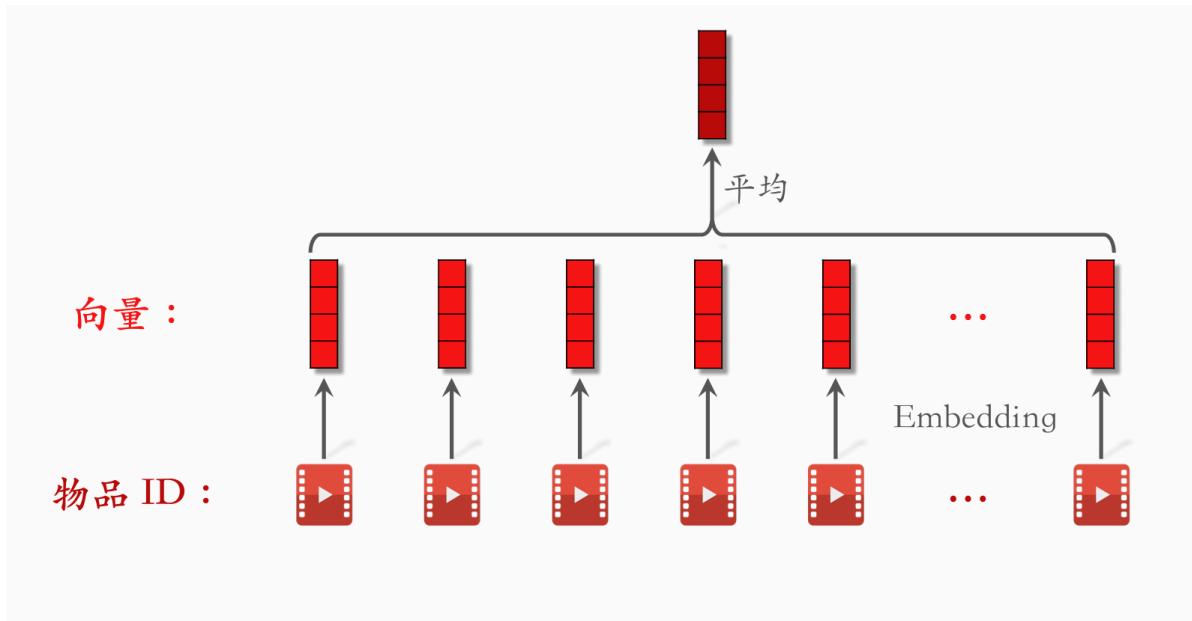
粗排模型

- 粗排的打分量比精排大 10 倍，因此粗排模型必须够快。
- **简单模型：**多向量双塔模型，同时预估点击率等多个目标。
- **复杂模型：**三塔模型效果好，但工程实现难度较大。

粗精排一致性建模

- 蒸馏精排训练粗排，让粗排与精排更一致。
- **方法1：**pointwise 蒸馏。
 - 设 y 是用户真实行为，设 p 是精排的预估。
 - 用 $\frac{y+p}{2}$ 作为粗排拟合的目标。
 - **例：**
 - 对于点击率目标，用户有点击 ($y = 1$)，精排预估 $p = 0.6$ 。
 - 用 $\frac{y+p}{2} = 0.8$ 作为粗排拟合的点击率目标。
- **方法2：**pairwise 或 listwise 蒸馏。
 - 给定 k 个候选物品，按照精排预估做排序。
 - 做 learning to rank (*LTR*)，让粗排拟合物品的序（而非值）。
 - **例：**
 - 对于物品 i 和 j ，精排预估点击率为 $p_i > p_j$ 。
 - *LTR* 鼓励粗排预估点击率满足 $q_i > q_j$ ，否则有惩罚。
 - *LTR* 通常使用 pairwise logistic loss。
- **优点：**粗精排一致性建模可以提升核心指标。
- **缺点：**如果精排出 bug，精排预估值 p 有偏，会污染粗排训练数据。

用户行为序列建模



- 最简单的方法是对物品向量取平均，作为一种用户特征。
- DIN* 使用注意力机制，对物品向量做加权平均。
- 工业界目前沿着 *SIM* 的方向发展。先用类别等属性筛选物品，然后用 *DIN* 对物品向量做加权平均。

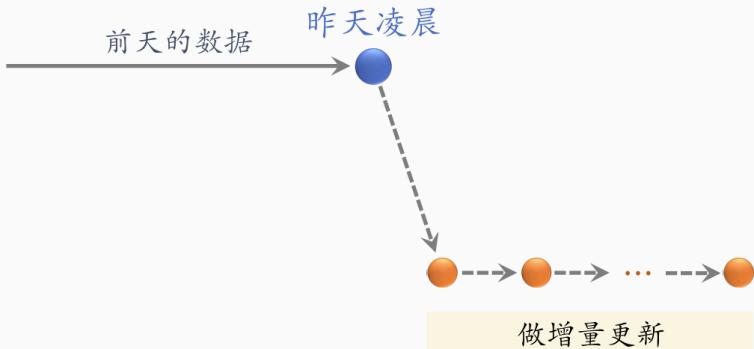
用户行为序列建模

- 改进1：**增加序列长度，让预测更准确，但是会增加计算成本和推理时间。
- 改进2：**筛选的方法，比如用类别、物品向量表征聚类。
 - 离线用多模态神经网络提取物品内容特征，将物品表征为向量。
 - 离线将物品向量聚类为 1000 类，每个物品有一个聚类序号。
 - 线上排序时，用户行为序列中有 $n = 1,000,000$ 个物品。某候选物品的聚类序号是 70，对 n 个物品做筛选，只保留聚类序号为 70 的物品。 n 个物品中只有数千个被保留下来。
 - 同时有好几种筛选方法，取筛选结果的并集。
- 改进3：**对用户行为序列中的物品，使用 ID 以外的一些特征。
- 概括：**沿着 *SIM* 的方向发展，让原始的序列尽量长，然后做筛选降低序列长度，最后将筛选结果输入 *DIN*。

在线学习

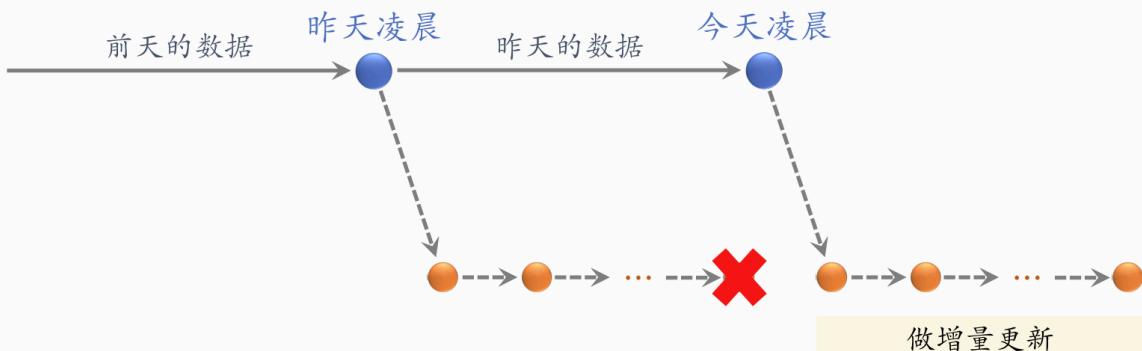
全量更新 vs 增量更新

基于前天的数据，用
前天的数据，做全量更新。



全量更新 vs 增量更新

基于昨天的数据，用
昨天的数据，做全量更新。



在线学习的资源消耗

- 既需要在凌晨做全量更新，也需要全天不断做增量更新。
- 设在线学习需要 $10,000 \text{ CPU core}$ 的算力增量更新一个精排模型。推荐系统一共需要多少额外的算力给在线学习？
- 为了做 AB 测试，线上同时运行多个不同的模型。
- 如果有 m 个模型，则需要 m 套在线学习的机器。
- 线上有 m 个模型，其中 1 个是 *holdout*，1 个是推全的模型， $m - 2$ 个测试的新模型。
- 每套在线学习的机器成本都很大，因此 m 数量很小，制约模型开发迭代的效率。
- 在线学习对指标的提升巨大，但是会制约模型开发迭代的效率。

在线学习的资源消耗



老汤模型

- 用每天新产生的数据对模型做 1 epoch 的训练。
 - 久而久之，老模型训练得非常好，很难被超过。
 - 对模型做改进，重新训练，很难追上老模型……
 - **问题 1：**如何快速判断新模型结构是否优于老模型？（不需要追上线上的老模型，只需要判断新老模型谁的结构更优。）
 - 对于新、老模型结构，都随机初始化模型全连接层。
 - *Embedding* 层可以是随机初始化，也可以是复用老模型训练好的参数。
 - 用 n 天的数据训练新老模型。（从旧到新，训练 1 epoch）
 - 如果新模型显著优于老模型，新模型很可能更优。
 - 只是比较新老模型结构谁更好，而非真正追平老模型。
 - **问题 2：**如何更快追平、超过线上的老模型？（只有几十天的数据，新模型就能追上训练上百天的老模型。）
 - 已经得出初步结论，认为新模型很可能优于老模型。用几十天的数据训练新模型，早日追平老模型。
 - **方法 1：**尽可能多地复用老模型训练好的 *embedding* 层，避免随机初始化 *embedding* 层。*(Embedding* 层是对用户、物品特征的“记忆”，比全连接层学得慢。)
 - **方法 2：**用老模型做 *teacher*，蒸馏新模型。（用户真实行为是 y ，老模型的预测是 p ，用 $\frac{y+p}{2}$ 作为训练新模型的目标。）

总结：改进排序模型

- **精排模型**: 改进模型基座（加宽加深、特征交叉、特征工程），改进多目标预估（增加新目标、 $MMoE$ 、*position bias*）。
 - **粗排模型**: 三塔模型（取代多向量双塔模型），粗精排一致性建模。
 - **用户行为序列建模**: 沿着 SIM 的方向迭代升级，加长序列长度，改进筛选物品的方法。
 - **在线学习**: 对指标提升大，但是会降低模型迭代升级效率。
 - **老汤模型** 制约模型迭代升级效率，需要特殊技巧。

涨指标的方法：提升多样性

排序的多样性

精排多样性

- 精排阶段，结合兴趣分数和多样性分数对物品 i 排序。
 - s_i : 兴趣分数，即融合点击率等多个预估目标。
 - d_i : 多样性分数，即物品 i 与已选中的物品的差异。
 - 用 $s_i + d_i$ 对物品做排序。
- 常用 MMR、DPP 等方法计算多样性分数，精排使用滑动窗口，粗排不使用滑动窗口。
 - 精排决定最终的曝光，曝光页面上邻近的物品相似度应该小。所以计算精排多样性要使用滑动窗口。
 - 粗排要考虑整体的多样性，而非一个滑动窗口中的多样性。
- 除了多样性分数，精排还使用打散策略增加多样性。
 - **类目**：当前选中物品 i ，之后 5 个位置不允许跟 i 的二级类目相同。
 - **多模态**：事先计算物品多模态内容向量表征，将全库物品聚为 1000 类；在精排阶段，如果当前选中物品 i ，之后 10 个位置不允许跟 i 同属一个聚类。

粗排多样性

- 粗排给 5000 个物品打分，选出 500 个物品送入精排。
- 提升粗排和精排多样性都可以提升推荐系统核心指标。
- 根据 s_i 对 5000 个物品排序，分数最高的 200 个物品送入精排。
- 对于剩余的 4800 个物品，对每个物品 i 计算兴趣分数 s_i 和多样性分数 d_i 。
- 根据 $s_i + d_i$ 对剩余 4800 个物品排序，分数最高的 300 个物品送入精排。

召回的多样性

双塔模型：添加噪声

- 用户塔将用户特征作为输入，输出用户的向量表征；然后做 ANN 检索，召回向量相似度高的物品。
- 线上做召回时（在计算出用户向量之后，在做 ANN 检索之前），往用户向量中添加随机噪声。
- 用户的兴趣越窄（比如用户最近交互的 n 个物品只覆盖少数几个类目），则添加的噪声越强。
- 添加噪声使得召回的物品更多样，可以提升推荐系统核心指标。

双塔模型：抽样用户行为序列

- 用户最近交互的 n 个物品（用户行为序列）是用户塔的输入。
- 保留最近的 r 个物品 ($r \ll n$)。
- 从剩余的 $n - r$ 个物品中随机抽样 t 个物品 ($t \ll n$)。（可以是均匀抽样，也可以用非均匀抽样让类目平衡。）
- 将得到的 $r + t$ 个物品作为用户行为序列，而不是用全部 n 个物品。
- **抽样用户行为序列为什么能涨指标？**
 - 一方面，注入随机性，召回结果更多样化。
 - 另一方面， n 可以非常大，可以利用到用户很久之前的兴趣。

U2I2I：抽样用户行为序列

- $U2I2I$ ($user \rightarrow item \rightarrow item$) 中的第一个 $item$ 是指用户最近交互的 n 个物品之一，在 $U2I2I$ 中叫作**种子物品**。
- n 个物品覆盖的类目数较少，且类目不平衡。
 - 系统共有 200 个类目，某用户的 n 个物品只覆盖 15 个类目。
 - 足球类目的物品有 $0.4n$ 个，电视剧类目的物品有 $0.2n$ 个，其余类目的物品数均少于 $0.05n$ 个。
- 做非均匀随机抽样，从 n 个物品中选出 t 个，让类目平衡。（想法和效果与双塔中的用户行为序列抽样相似。）
- 用抽样得到的 t 个物品（代替原本的 n 个物品）作为 $U2I2I$ 的种子物品。
- 一方面，类目更平衡，多样性更好。另一方面， n 可以更大，覆盖的类目更多。

探索流量

- 每个用户曝光的物品中有 2% 是非个性化的，用作兴趣探索。
- 维护一个精选内容池，其中物品均为交互率指标高的优质物品。（内容池可以分人群，比如 30~40 岁男性内容池。）
- 从精选内容池中随机抽样几个物品，跳过排序，直接插入最终排序结果。
- 兴趣探索在短期内负向影响核心指标，但长期会产生正向影响。

总结：提升多样性

- **精排**：结合兴趣分数和多样性分数做排序；做规则打散。
- **粗排**：只用兴趣分数选出部分物品；结合兴趣分数和多样性分数选出部分物品。
- **召回**：往双塔模型的用户向量添加噪声；对用户行为序列做非均匀随机抽样（对双塔和 U2I2I 都适用）。
- **兴趣探索**：保留少部分的流量给非个性化推荐。

涨指标的方法：特殊对待特殊人群

为什么要特殊对待特殊人群？

1. 新用户、低活用户的行为很少，个性化推荐不准确。
2. 新用户、低活用户容易流失，要想办法促使他们留存。
3. 特殊用户的行为（比如点击率、交互率）不同于主流用户，基于全体用户行为训练出的模型在特殊用户人群上有偏。

涨指标的方法

1. 构造特殊内容池，用于特殊用户人群的召回。
2. 使用特殊排序策略，保护特殊用户。
3. 使用特殊的排序模型，消除模型预估的偏差。

构造特殊的内容池

特殊内容池

- 为什么需要特殊内容池？
- 新用户、低活用户的行为很少，个性化召回不准确。（既然个性化不好，那么就保证内容质量好。）

- 针对特定人群的特点构造特殊内容池，提升用户满意度。例如，对于喜欢留下评论的中年女性，构造促进评论内容池，满足这些用户的互动需求。

如何构造特殊内容池

- 方法 1：根据物品获得的交互次数、交互率选择优质物品。
 - 圈定人群：只考虑特定人群，例如 18~25 岁二线城市男性。
 - 构造内容池：用该人群对物品的交互次数、交互率给物品打分，选出分数最高的物品进入内容池。
 - 内容池有弱个性化的效果。
 - 内容池定期更新，加入新物品，排除交互率低和失去时效性的老物品。
 - 该内容池只对该人群生效。
- 方法 2：做因果推断，判断物品对人群留存率的贡献，根据贡献值选物品。

特殊内容池的召回

- 通常使用双塔模型从特殊内容池中做召回。
 - 双塔模型是个性化的。
 - 对于新用户，双塔模型的个性化做不准。
 - 靠高质量内容、弱个性化做弥补。
- 额外的训练代价？
 - 对于正常用户，不论有多少内容池，只训练一个双塔模型。
 - 对于新用户，由于历史交互记录很少，需要单独训练模型。
- 额外的推理代价？
 - 内容池定期更新，然后需要更新 ANN 索引。
 - 线上做召回时，需要做 ANN 检索。
 - 特殊内容池都很小（比全量内容池小 10~100 倍），所以需要的额外算力不大。

特殊的排序策略

差异化的排序模型

- 特殊用户人群的行为不同于普通用户。新用户、低活用户的点击率、交互率偏高或偏低。
- 排序模型被主流用户主导，对特殊用户做不准预估。
 - 用全体用户数据训练出的模型，给新用户做的预估有严重偏差。
 - 如果一个APP的用90%是女性，用全体用户数据训练出的模型，对男性用户做的预估有偏差。
- 问题：对于特殊用户，如何让排序模型预估做得准？
- **方法 1：大模型 + 小模型。**
 - 用全体用户行为训练大模型，大模型的预估 p 拟合用户行为 y 。
 - 用特殊用户的行为训练小模型，小模型的预估 q 拟合大模型的残差 $y - p$ 。
 - 对主流用户只用大模型做预估 p 。
 - 对特殊用户，结合大模型和小模型的预估 $p + q$ 。
- **方法 2：融合多个 experts，类似 MMoE。**
 - 只用一个模型，模型有多个 experts，各输出一个向量。

- 对 experts 的输出做加权平均。
- 根据用户特征计算权重。
- 以新用户为例，模型将用户的新老、活跃度等特征作为输入，输出权重，用于对 experts 做加权平均。
- **方法 3：大模型预估之后，用小模型做校准。**
 - 用大模型预估点击率、交互率。
 - 将用户特征、大模型预估点击率和交互率作为小模型（例如 GBDT）的输入。
 - 在特殊用户人群的数据上训练小模型，小模型的输出拟合用户真实行为。

错误的做法

- 每个用户人群使用一个排序模型，推荐系统同时维护多个大模型。
 - 系统有一个主模型；每个用户人群有自己的一个模型。
 - 每天凌晨，用全体用户数据更新主模型。
 - 基于训练好的主模型，在某特殊用户人群的数据上再训练 1 epoch，作为该用户人群的模型。
- 短期可以提升指标；维护代价大，长期有害。
 - 起初，低活男性用户模型比主模型的 AUC 高 0.2%。
 - 主模型迭代几个版本后，AUC 累计提升 0.5%。
 - 特殊人群模型太多，长期没有人维护和更新。
 - 如果把低活男性用户模型下线，换成主模型，在低活男性用户上的 AUC 反倒提升 0.3%！

总结：特殊对待特殊用户人群

- **召回：**针对特殊用户人群，构造特殊的内容池，增加相应的召回通道。
- **排序策略：**排除低质量物品，保护新用户和低活用户；特殊用户人群使用特殊的融分公式。
- **排序模型：**结合大模型和小模型，小模型拟合大模型的残差；只用一个模型，模型有多个 experts；大模型预估之后，用小模型做校准。

涨指标的方法：利用交互行为

关注

关注量对留存的价值

- 对于一位用户，他关注的作者越多，则平台对他的吸引力越强。
- 用户留存率 (r) 与他关注的作者数量 (f) 正相关。
- 如果某用户的 f 较小，则推荐系统要促使该用户关注更多作者。
- 如何利用关注关系提升用户留存？
 - 方法 1：用排序策略提升关注量。
 - 对于用户 u ，模型预估候选物品 i 的关注率为 p_i 。
 - 设用户 u 已经关注了 f 个作者。
 - 我们定义单调递减函数 $w(f)$ ，用户已经关注的作者越多，则 $w(f)$ 越小。
 - 在排序融分公式中添加 $w(f) \cdot p_i$ ，用于促关注。（如果 f 小且 p_i 大，则 $w(f) \cdot p_i$ 给物品 i 带来很大加分。）
 - 方法 2：构造促关注内容池和召回通道。

- 这个内容池中物品的关注率高，可以促关注。
- 如果用户关注的作者数 f 较小，则对该用户使用该内容池。
- 召回配额可以固定，也可以与 f 负相关。

粉丝数对促发布的价值

- UGC 平台将作者发布量、发布率作为核心指标，希望作者多发布。
- 作者发布的物品被平台推送给用户，会产生点赞、评论、关注等交互。
- 交互（尤其是关注、评论）可以提升作者发布积极性。
- 作者的粉丝数越少，则每增加一个粉丝对发布积极性的提升越大。
- 用排序策略帮助低粉新作者涨粉。
- 某作者 a 的粉丝数（被关注数）为 f_a 。
- 作者 a 发布的物品 i 可能被推荐给用户 u ，模型预估关注率为 p_{ui} 。
- 我们定义单调递减函数 $w(f_a)$ 作为权重；作者 a 的粉丝越多，则 $w(f_a)$ 越小。
- 在排序融分公式中添加 $w(f_a) \cdot p_{ui}$ ，帮助低粉作者涨粉。

隐式关注关系

- **召回通道 U2A2I**: user → author → item。
- **显式关注关系**: 用户 u 关注了作者 a ，将 a 发布的物品推荐给 u 。（点击率、交互率指标通常高于其他召回通道。）
- **隐式关注关系**: 用户 u 喜欢看作者 a 发布的物品，但是 u 没有关注 a 。
- **隐式关注的作者数量远大于显式关注**。挖掘隐式关注关系，构造 U2A2I 召回通道，可以提升推荐系统核心指标。

转发（分享）

促转发（分享回流）

- A 平台用户将物品转发到 B 平台，可以为 A 吸引站外流量。
- 推荐系统做促转发（也叫分享回流）可以提升 DAU 和消费指标。
- 简单提升转发次数是否有效呢？
 - 模型预估转发率为 p ，融分公式中有一项 $w \cdot p$ ，让转发率大的物品更容易获得曝光机会。
 - 增大权重 w 可以促转发，吸引站外流量，但是会负面影响点击率和其他交互率。

KOL 建模

- 目标：在不损害点击和其他交互的前提下，尽量多吸引站外流量。
- 什么样的用户的转发可以吸引大量站外流量？其他平台的 Key Opinion Leader (KOL)！
- 如何判断本平台的用户是不是其他平台的 KOL？
- 该用户历史上的转发能带来多少站外流量。
- 方法2：构造促转发内容池和召回通道，对站外 KOL 生效。

评论

评论促发布

- UGC 平台 将作者发布量、发布率作为核心指标，希望作者多发布。
- 关注、评论等交互 可以提升作者发布积极性。
- 如果新发布物品尚未获得很多评论，则给预估评论率提权，让物品尽快获得评论。
- 排序融分公式中添加额外一项 $w_i \cdot p_i$ 。
 - w_i : 权重，与物品 i 已有的评论数量负相关。
 - p_i : 为用户推荐物品 i ，模型预估的评论率。

评论的其他价值

- 有的用户喜欢留言评论，喜欢跟作者、评论区用户互动。
 - 给这样的用户添加促评论的内容池，让他们更多机会参与讨论。
 - 有利于提升这些用户的留存。
- 有的用户常留高质量评论（评论的点赞量高）。
 - 高质量评论对作者、其他用户的留存有贡献。（作者、其他用户觉得这样的评论有趣或者有帮助。）
 - 用排序和召回策略鼓励这些用户多留评论。

总结：利用交互行为

- **关注：**
 - 留存价值（让新用户关注更多作者，提升新用户留存）。
 - 发布价值（帮助新作者获得更多粉丝，提升作者发布积极性）。
 - 利用隐式关注关系做召回。
- **转发：**判断哪些用户是站外的 KOL，利用他们转发的价值，吸引站外的流量。
- **评论：**
 - 发布价值（促使新物品获得评论，提升作者发布积极性）。
 - 留存价值（给喜欢讨论的用户创造更多留言机会）。
 - 鼓励高质量评论的用户多留评论。