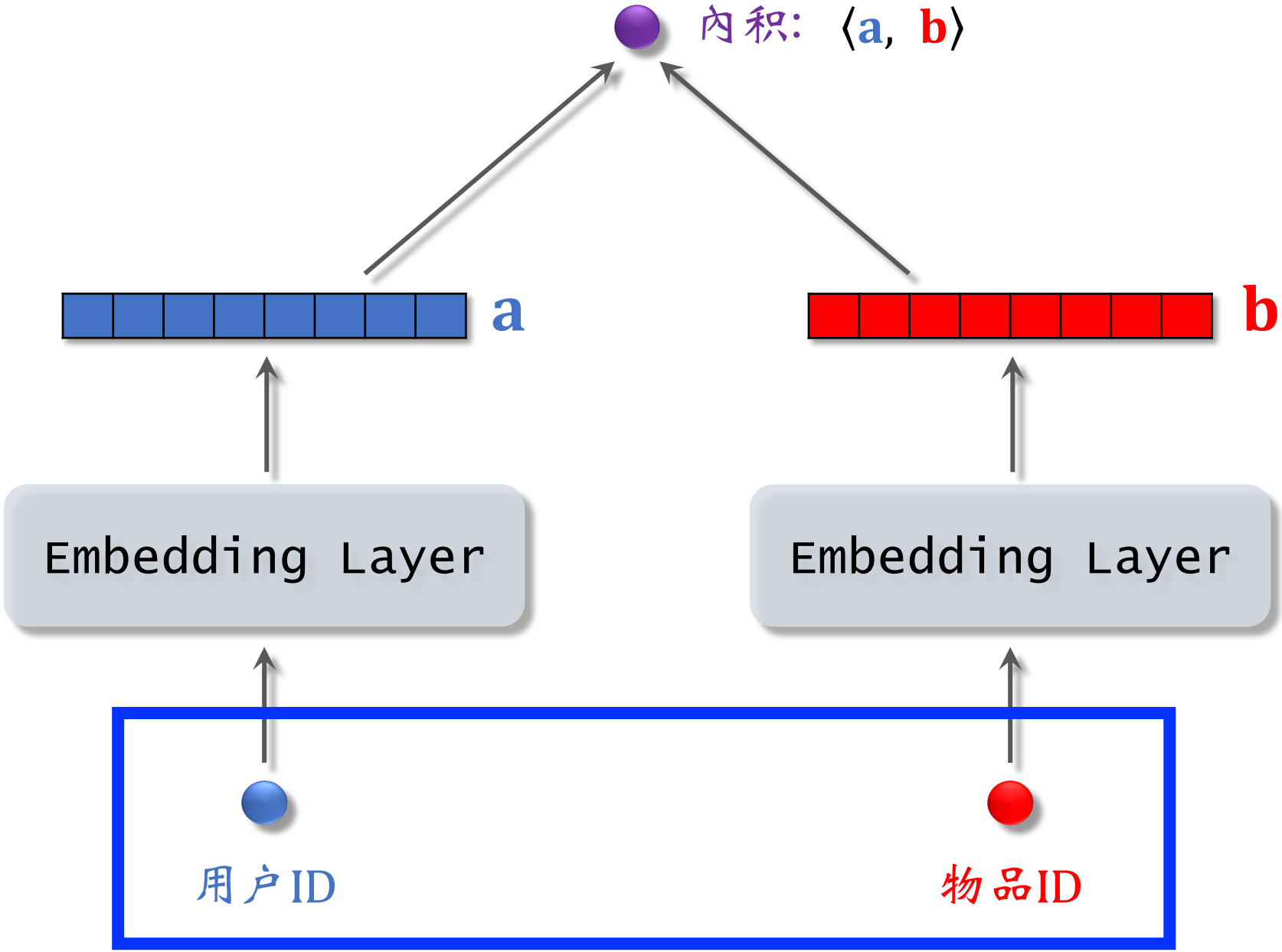
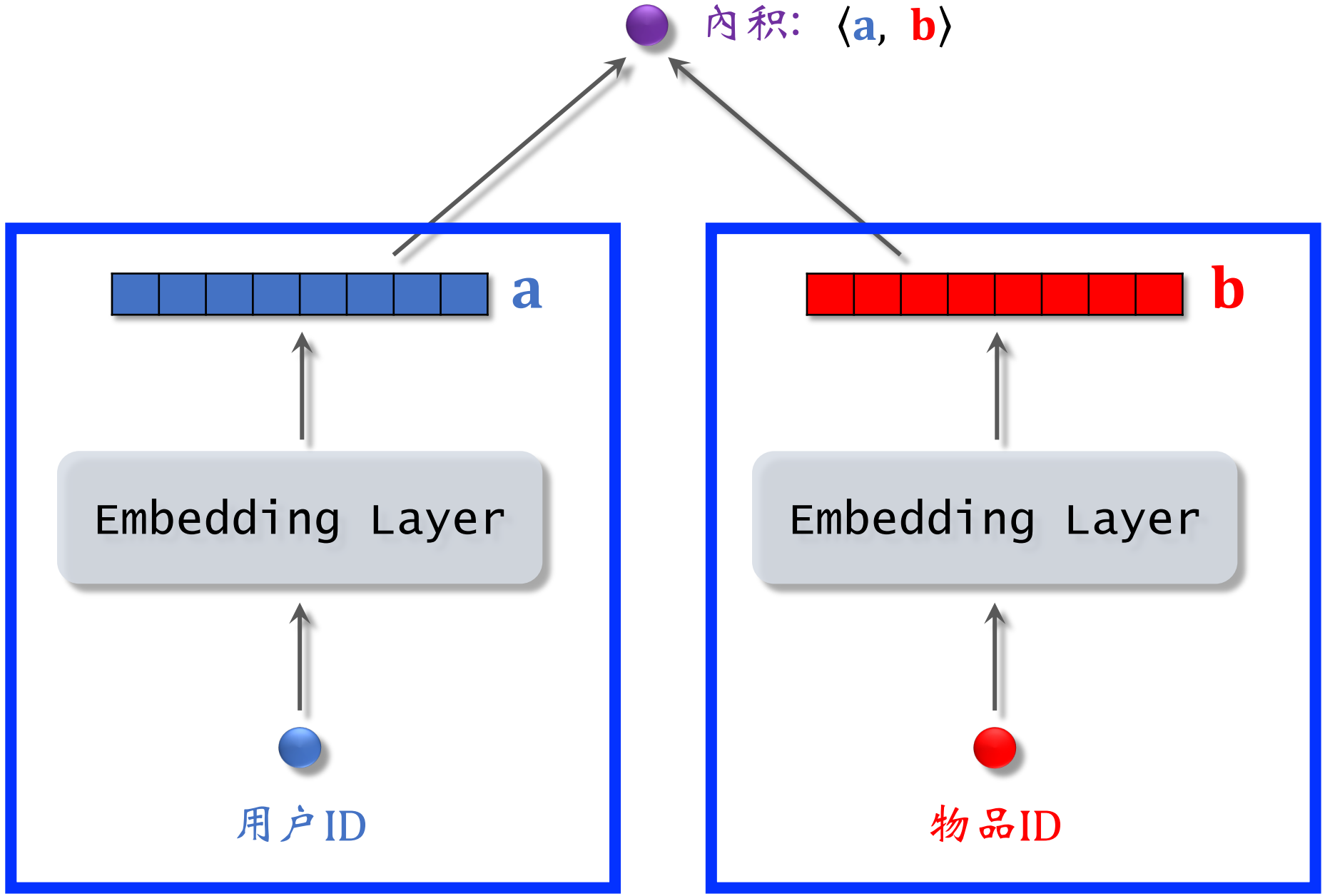
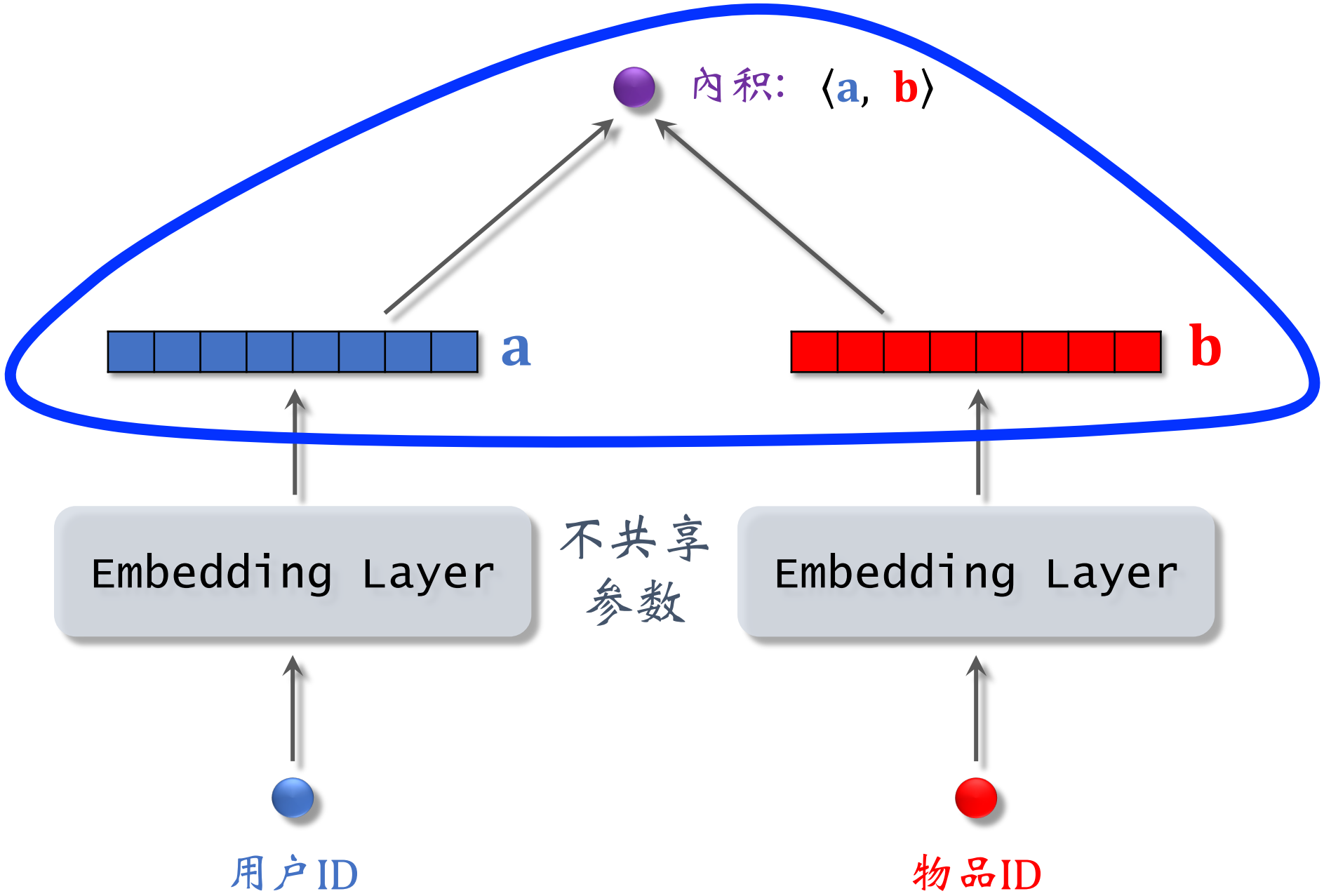
矩阵补充
(Matrix Completion)

王树森

http://wangshusen.github.io/


内积: ⟨a, b⟩
a
b
Embedding Layer
Embedding Layer
用户ID
物品ID

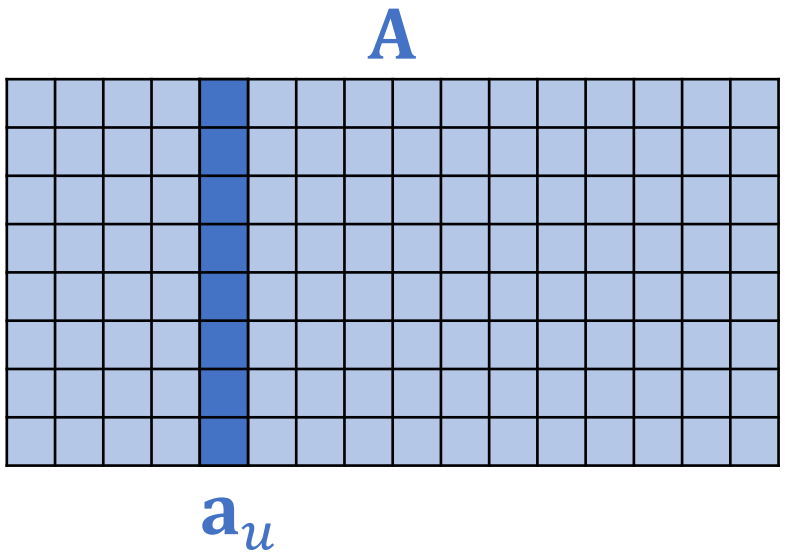

内积: ⟨a, b⟩
a
b
Embedding Layer
Embedding Layer
用户ID
物品ID

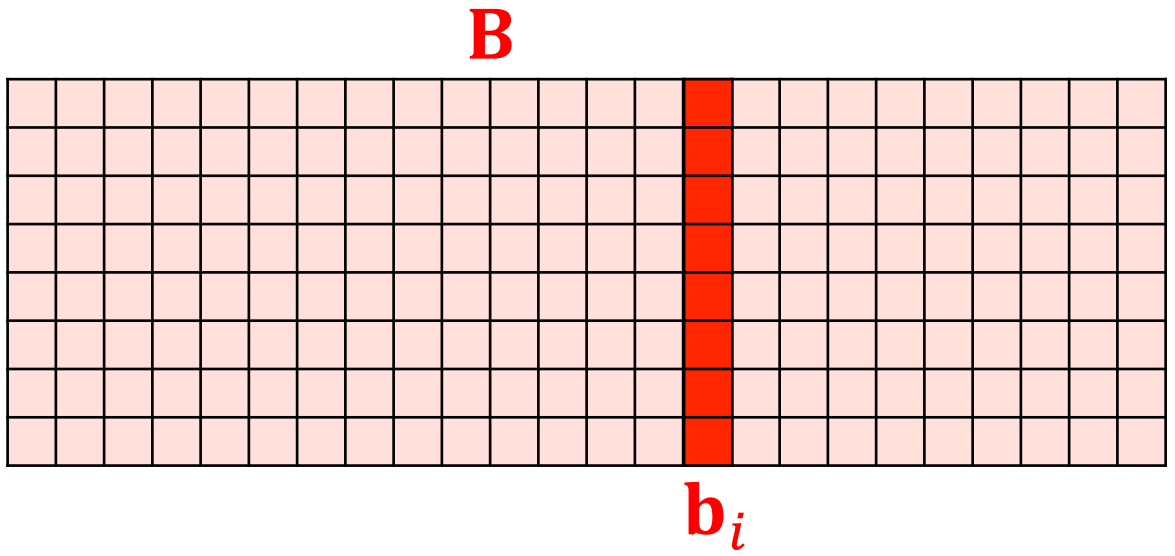

内积: ⟨a, b⟩
a
b
Embedding Layer
不共享
参数
Embedding Layer
用户ID
物品ID

训练

基本想法

- 用户 embedding 参数矩阵记作 $\mathbf{A}$。第 u 号用户对应矩阵第 u 列，记作向量 $\mathbf{a}_u$ 。
- 物品 embedding 参数矩阵记作 $\mathbf{B}$。第 i 号物品对应矩阵第 i 列，记作向量 $\mathbf{b}_i$ 。





基本想法

- 用户 embedding 参数矩阵记作 $\mathbf{A}$。第 u 号用户对应矩阵第 u 列，记作向量 $\mathbf{a}_u$ 。
- 物品 embedding 参数矩阵记作 $\mathbf{B}$。第 i 号物品对应矩阵第 i 列，记作向量 $\mathbf{b}_i$ 。
- 内积 $\langle \mathbf{a}_u, \mathbf{b}_i \rangle$ 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 $\mathbf{A}$ 和 $\mathbf{B}$，使得预估值拟合真实观测的兴趣分数。

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，比如：
  - 曝光但是没有点击 → 0 分
  - 点击、点赞、收藏、转发 → 各算 1 分
  - 分数最低是 0，最高是 4。

训练

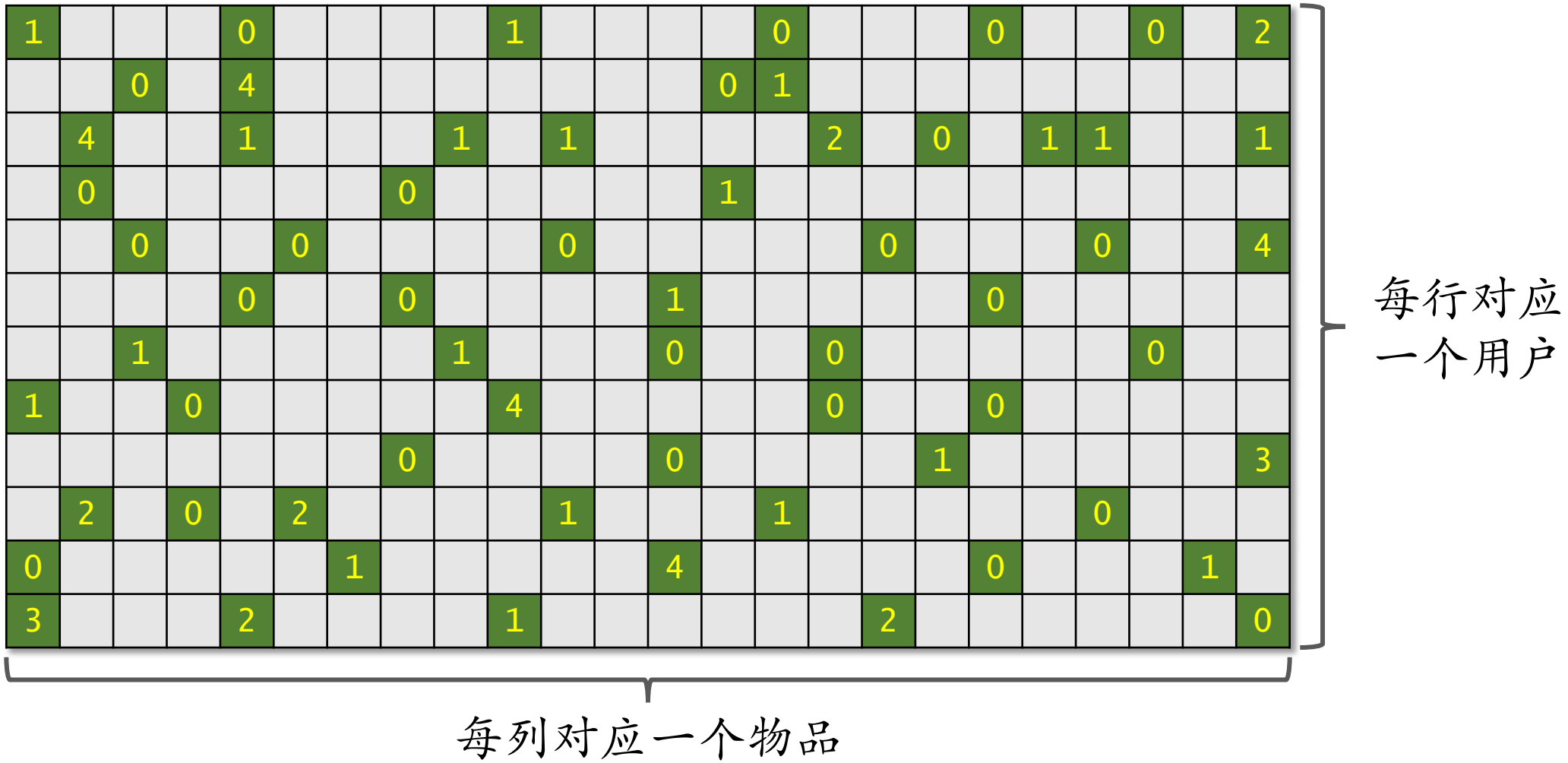
- 把用户ID、物品ID映射成向量。
  - 第 u 号用户 → 向量 $\mathbf{a}_u$。
  - 第 i 号物品 → 向量 $\mathbf{b}_i$。

训练

- 把用户ID、物品ID映射成向量。
  - 第 u 号用户 → 向量 $\mathbf{a}_u$。
  - 第 i 号物品 → 向量 $\mathbf{b}_i$。
- 求解优化问题，得到参数 $\mathbf{A}$ 和 $\mathbf{B}$ 。

$$\min_{\mathbf{A},\mathbf{B}} \sum_{(u,i,y)\in\Omega} \left(y - \langle \mathbf{a}_u, \mathbf{b}_i \rangle \right)^2 .$$

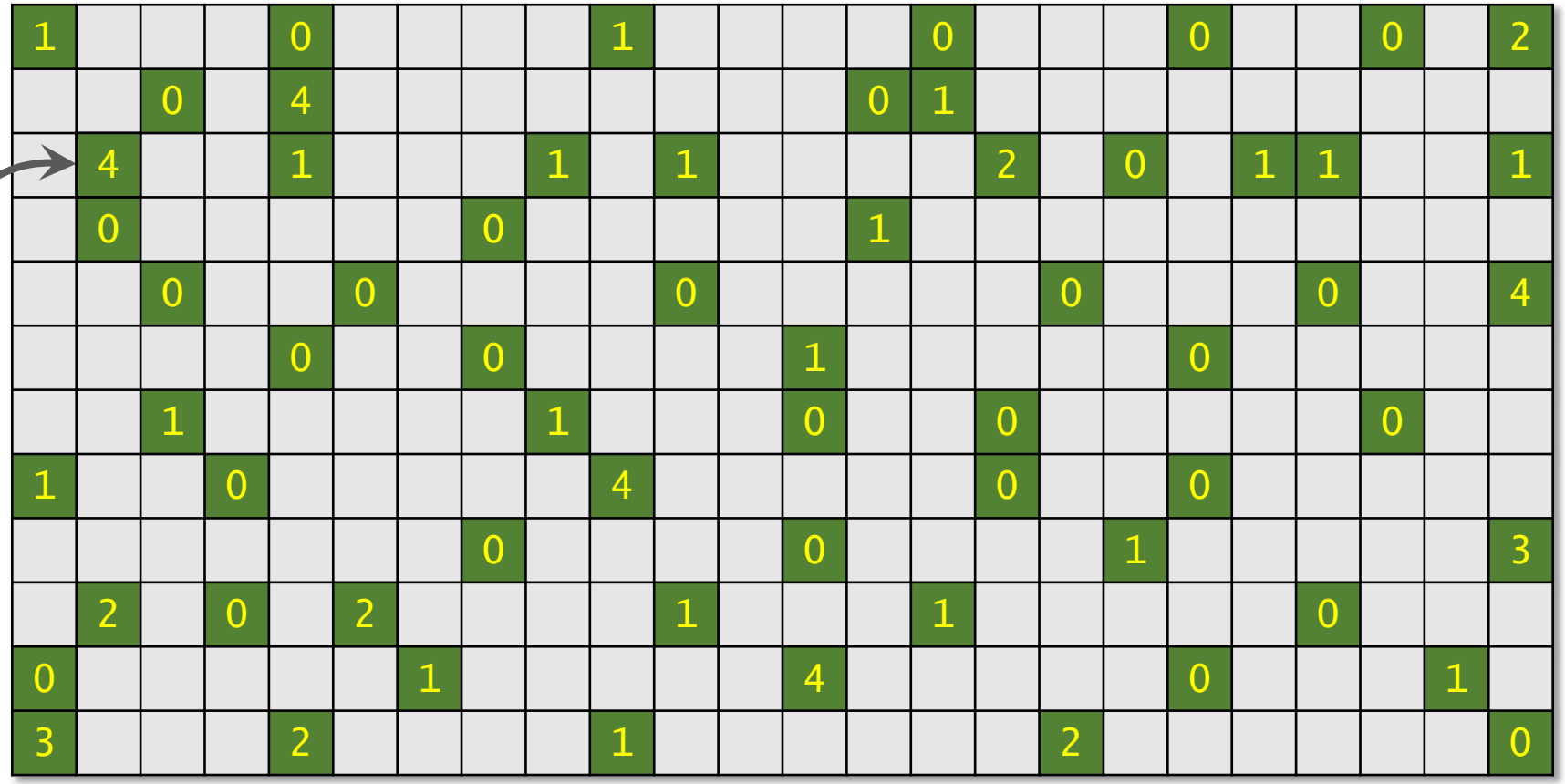
矩阵补充



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



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

在实践中效果不好……

缺点1：仅用ID embedding，没利用物品、用户属性。

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）

在实践中效果不好……

缺点1：仅用ID embedding，没利用物品、用户属性。

缺点2：负样本的选取方式不对。

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）。

线上服务

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。

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**。

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 $\mathbf{a}$。
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。

线上服务

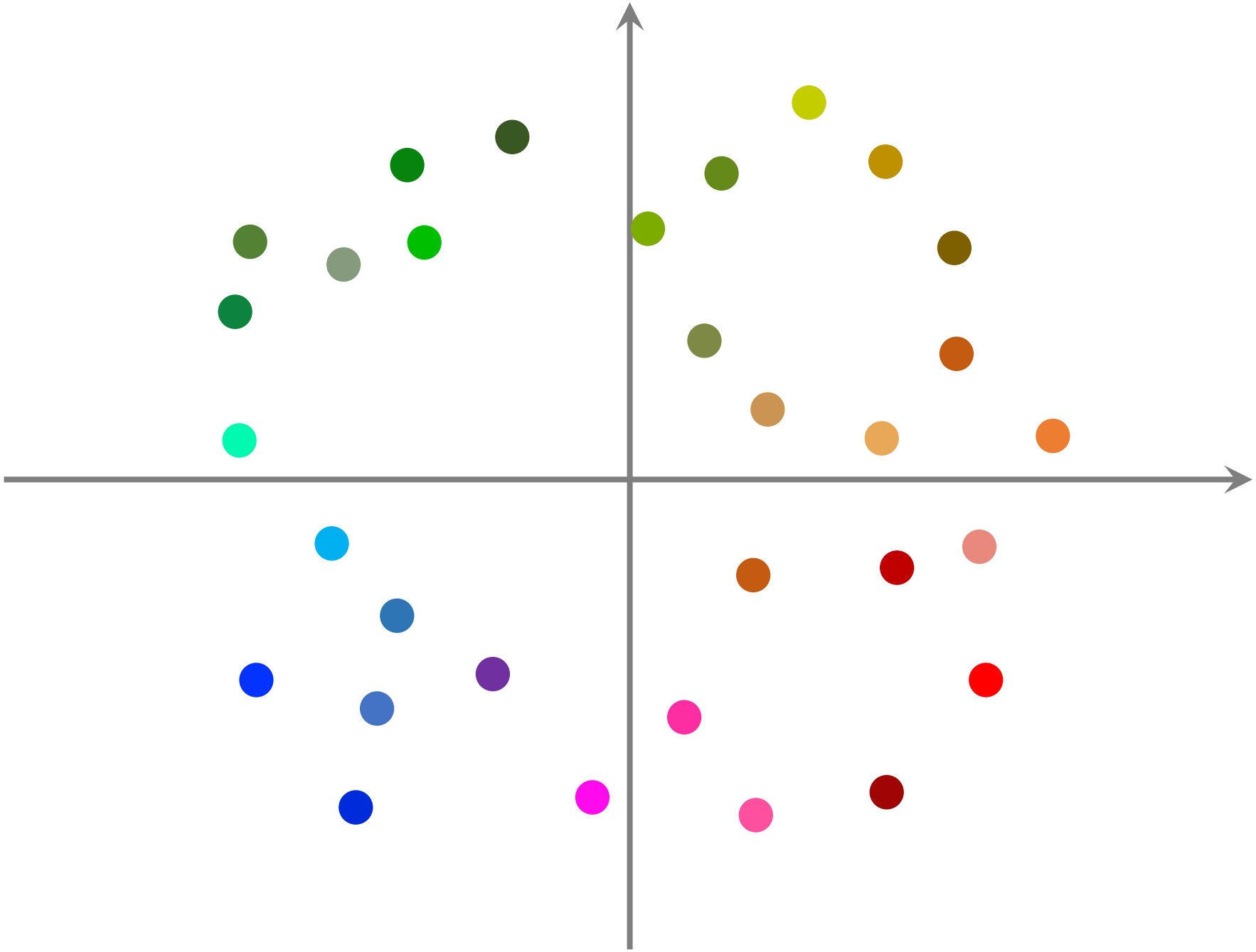
1. 把用户 ID 作为 key，查询 key-value 表，得到该用户的向量，记作 $\mathbf{a}$。
2. 最近邻查找：查找用户最有可能感兴趣的 k 个物品，作为召回结果。
   - 第 i 号物品的 embedding 向量记作 $\mathbf{b}_i$。
   - 内积 $\langle \mathbf{a}, \mathbf{b}_i \rangle$ 是用户对第 i 号物品兴趣的预估。
   - 返回内积最大的 k 个物品。

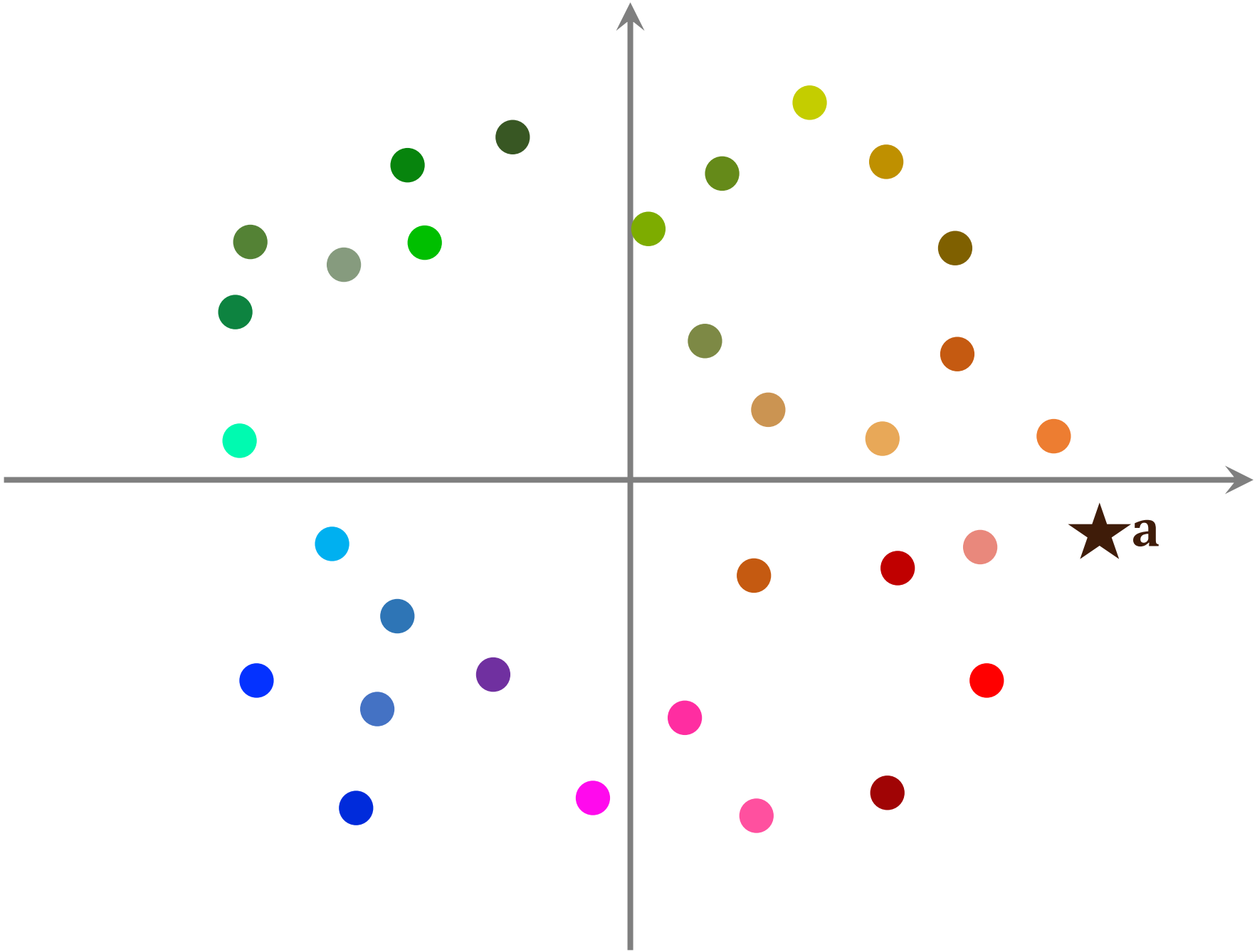
如果枚举所有物品，时间复杂度正比于物品数量。

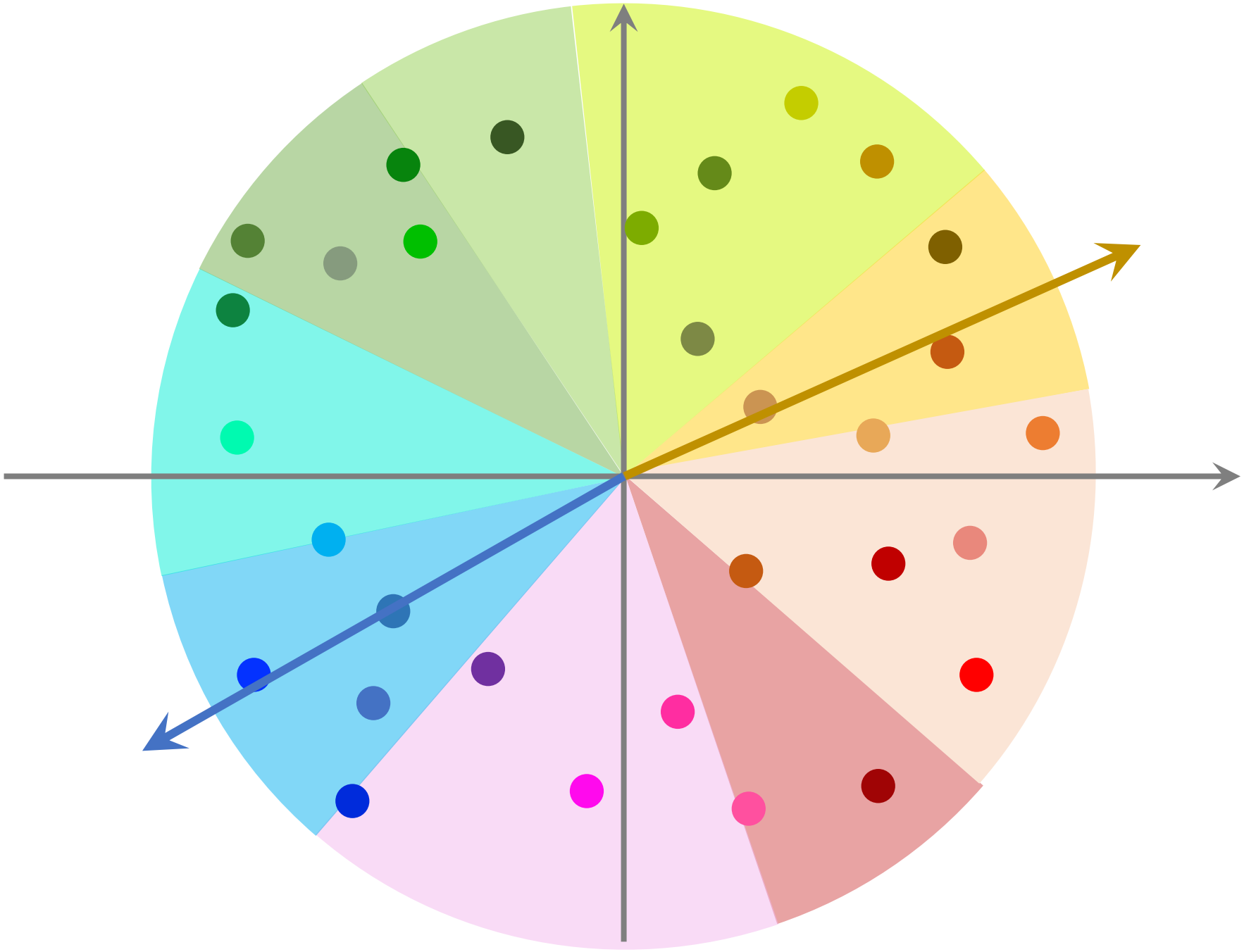
近似最近邻查找
(Approximate Nearest Neighbor Search)

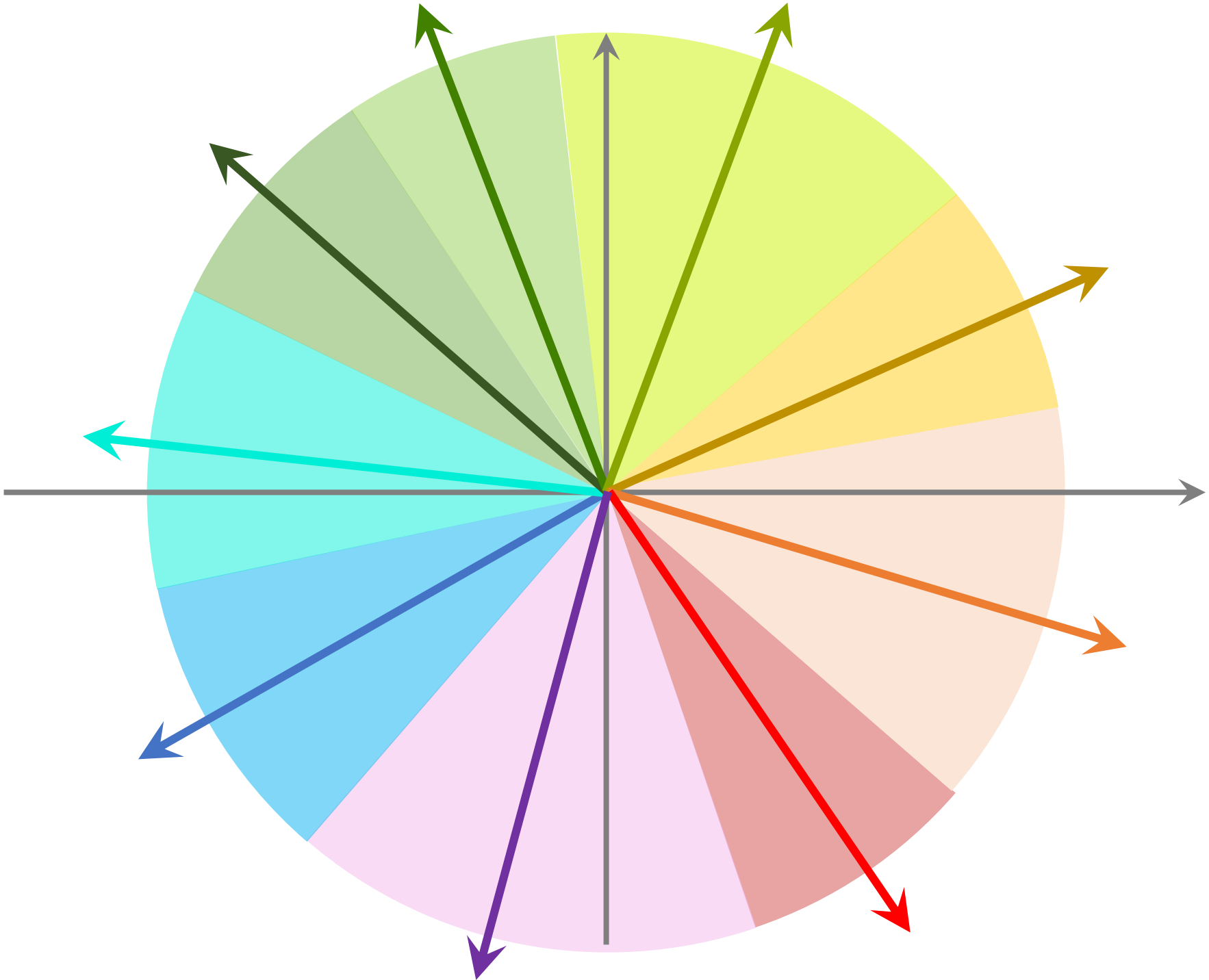
支持最近邻查找的系统

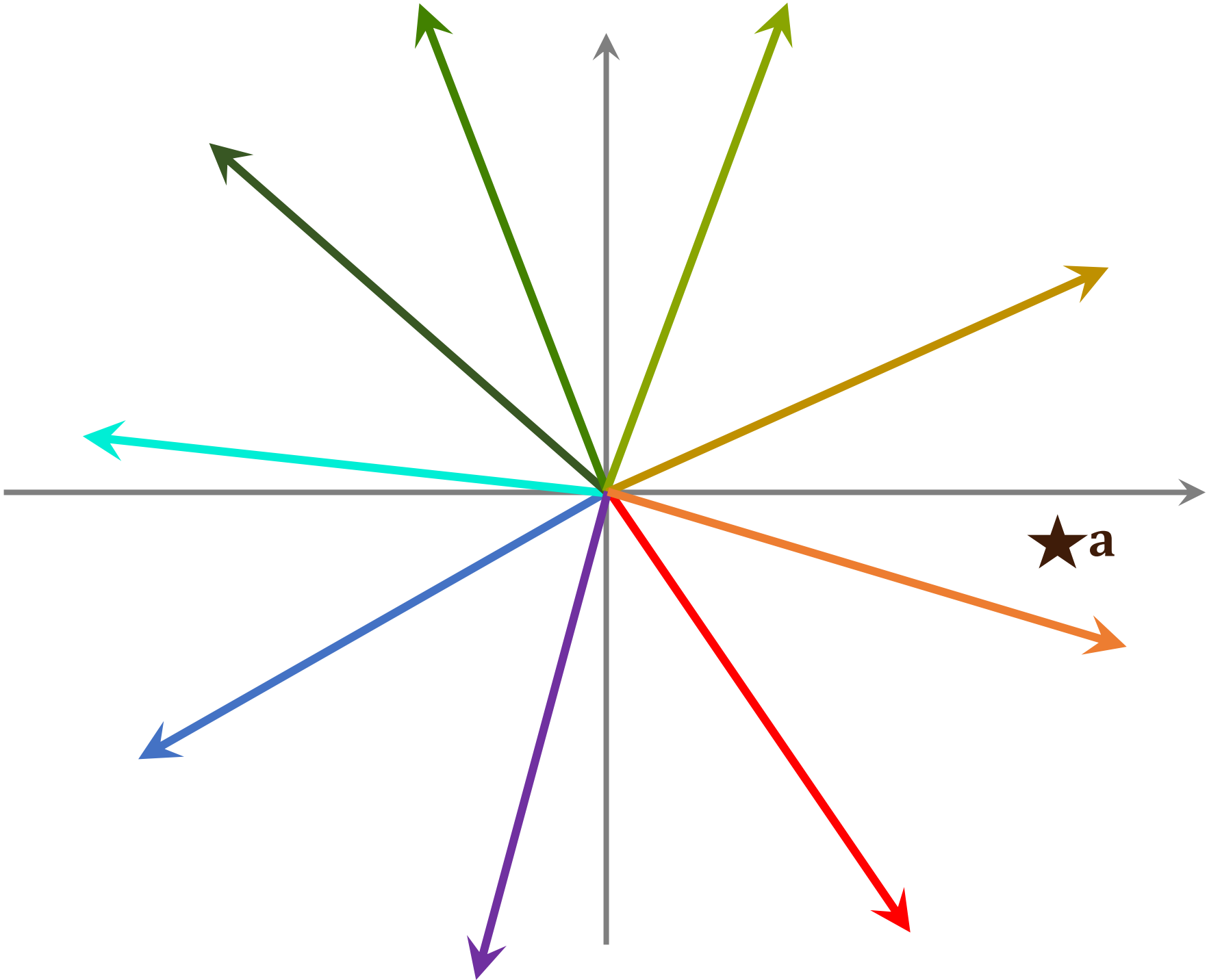
- **系统：**Milvus、Faiss、HnswLib、等等。
- 衡量最近邻的标准：
  - 欧式距离最小（L2 距离）
  - 向量内积最大（内积相似度）
  - 向量夹角余弦最大（cosine相似度）

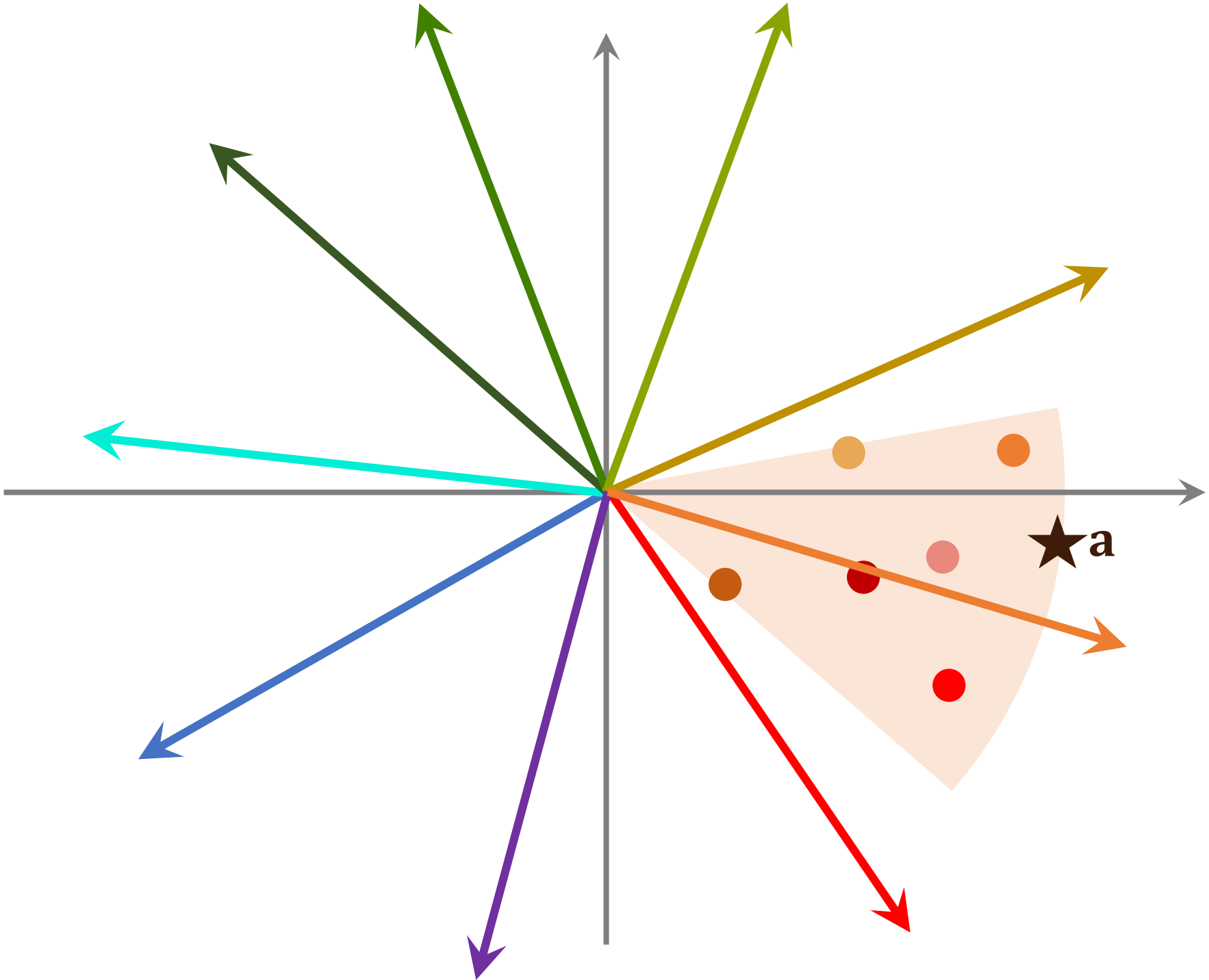


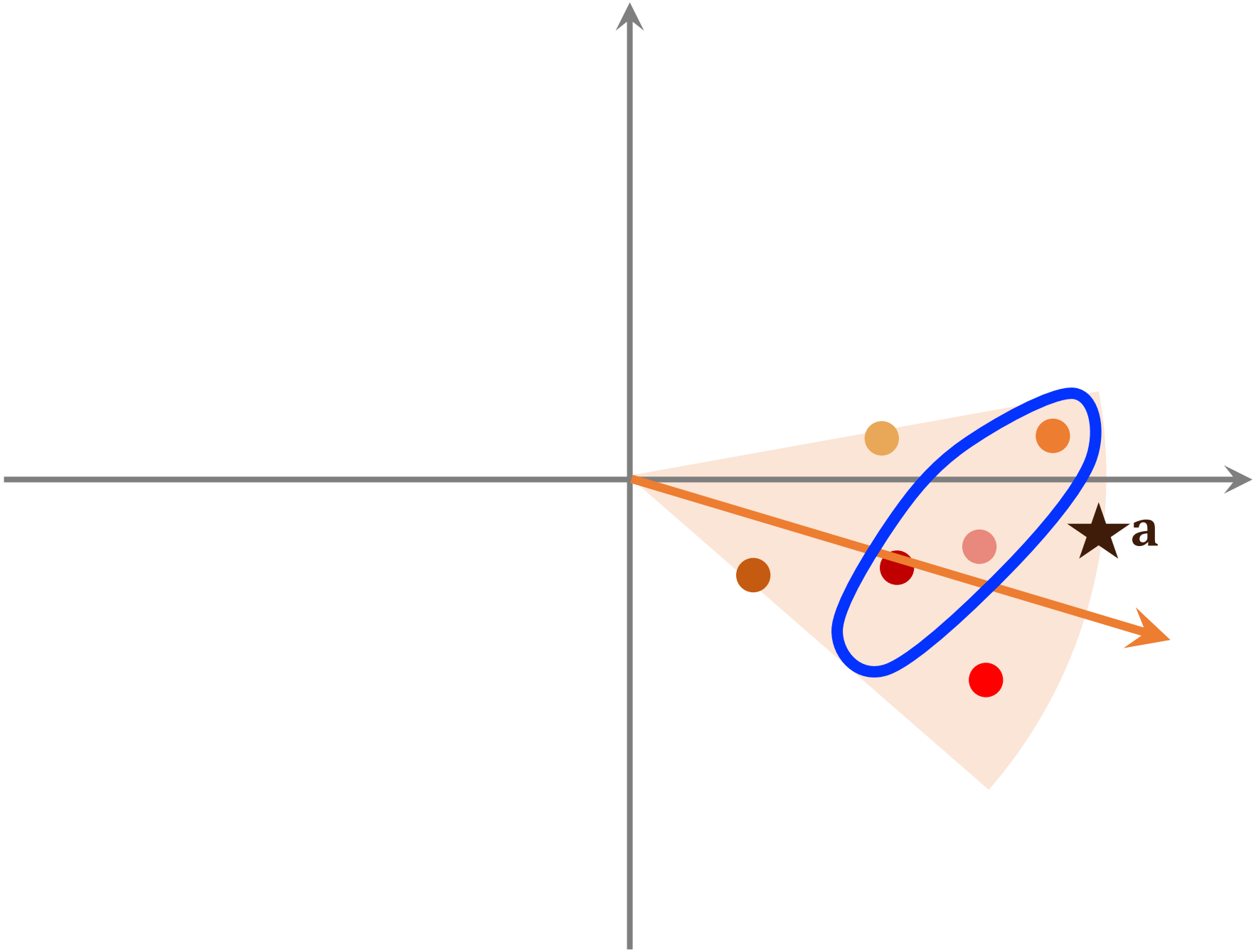

a






a


a


a

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。

Thank You!

http://wangshusen.github.io/