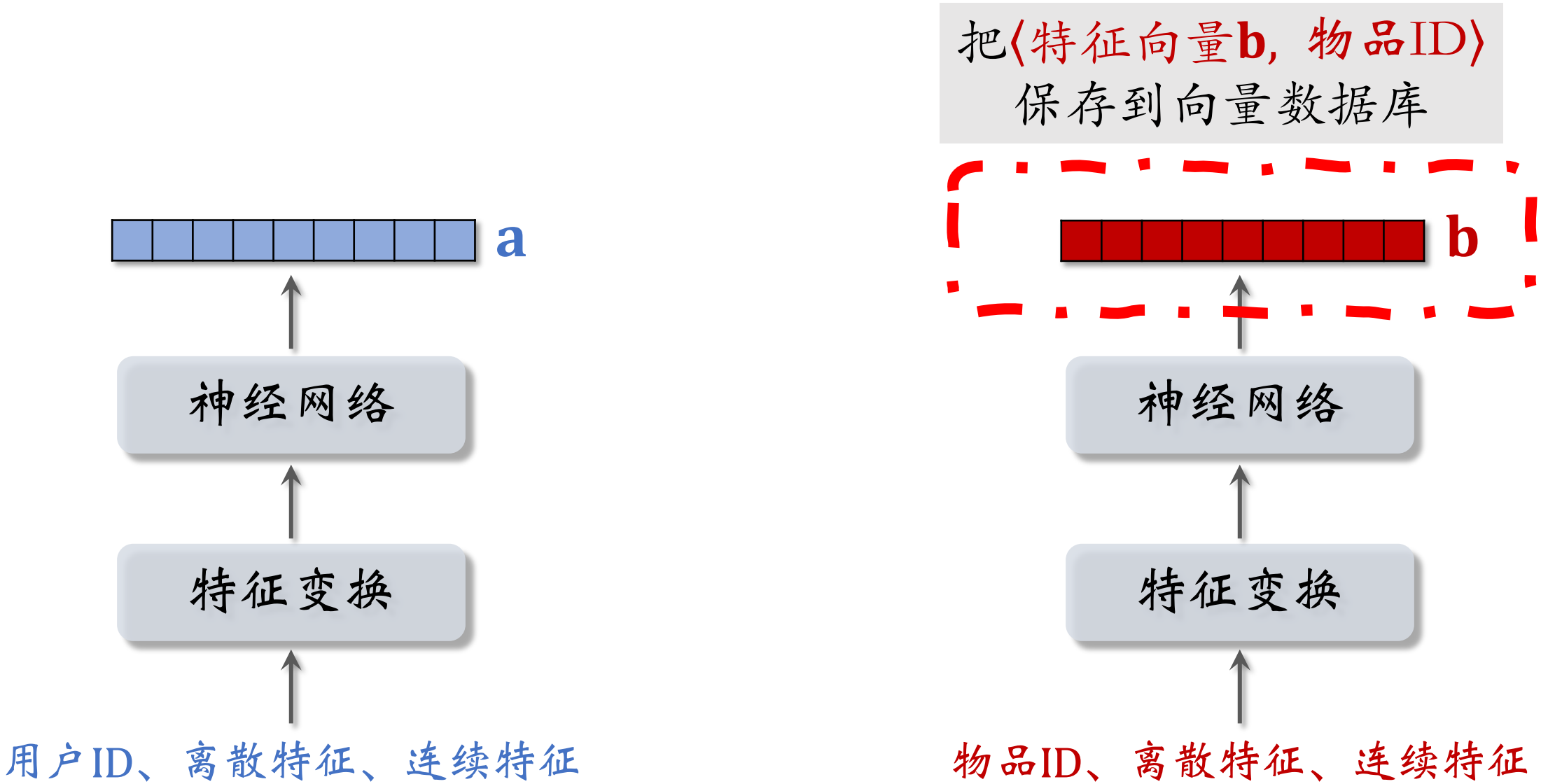
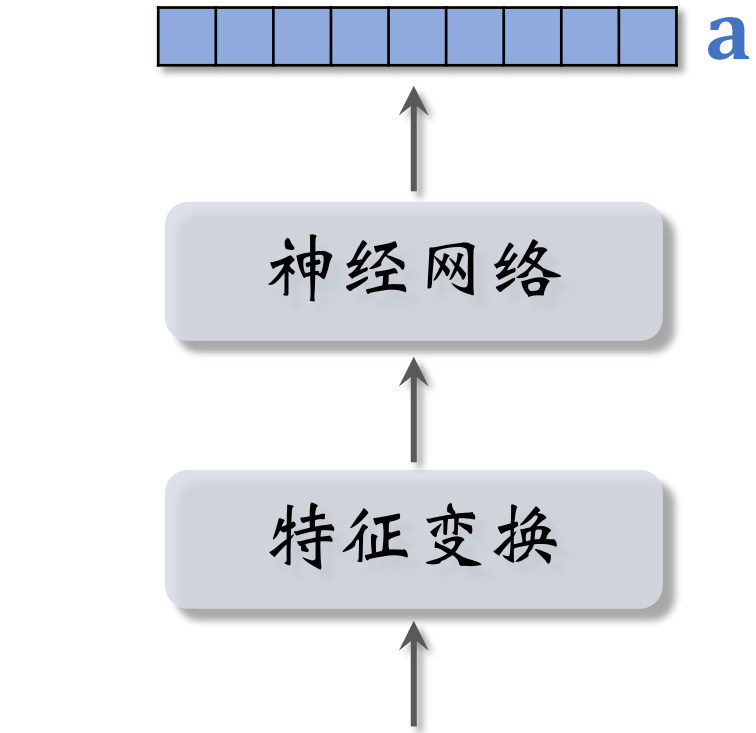
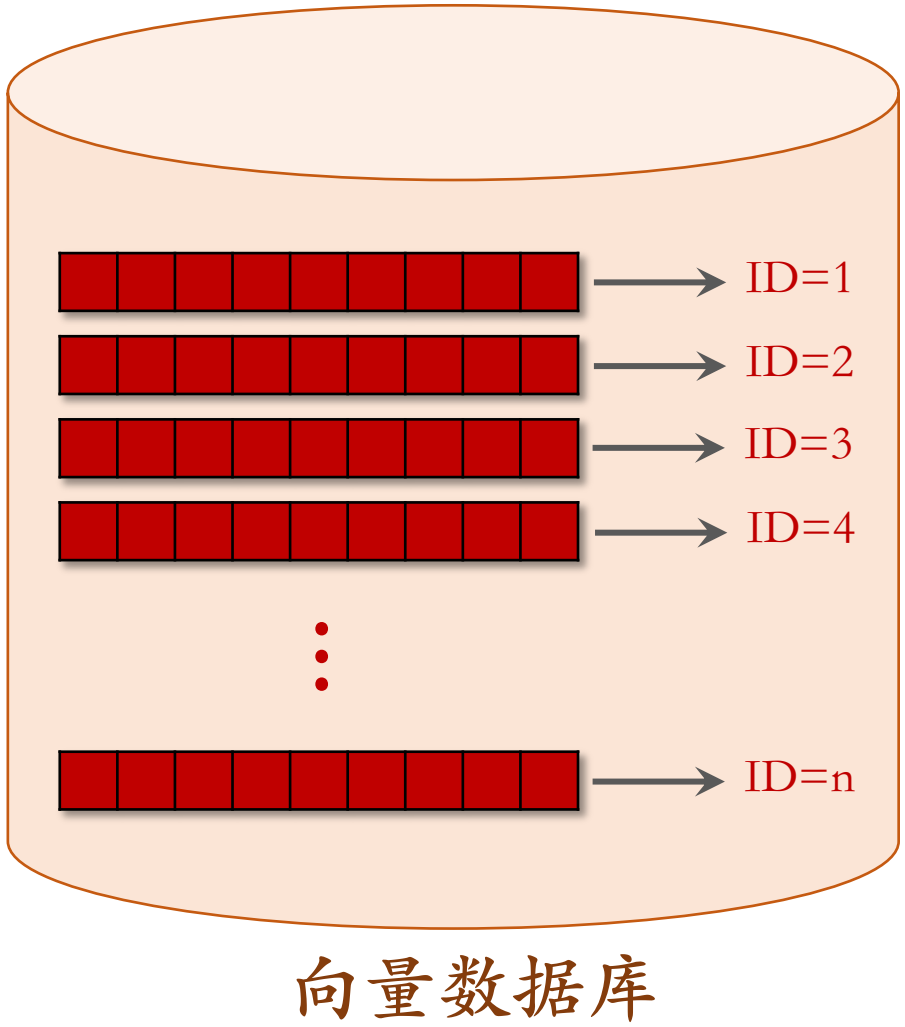
线上召回

离线存储


把⟨特征向量b，物品ID⟩
保存到向量数据库
a
b
神经网络
神经网络
特征变换
特征变换
用户ID、离散特征、连续特征
物品ID、离散特征、连续特征

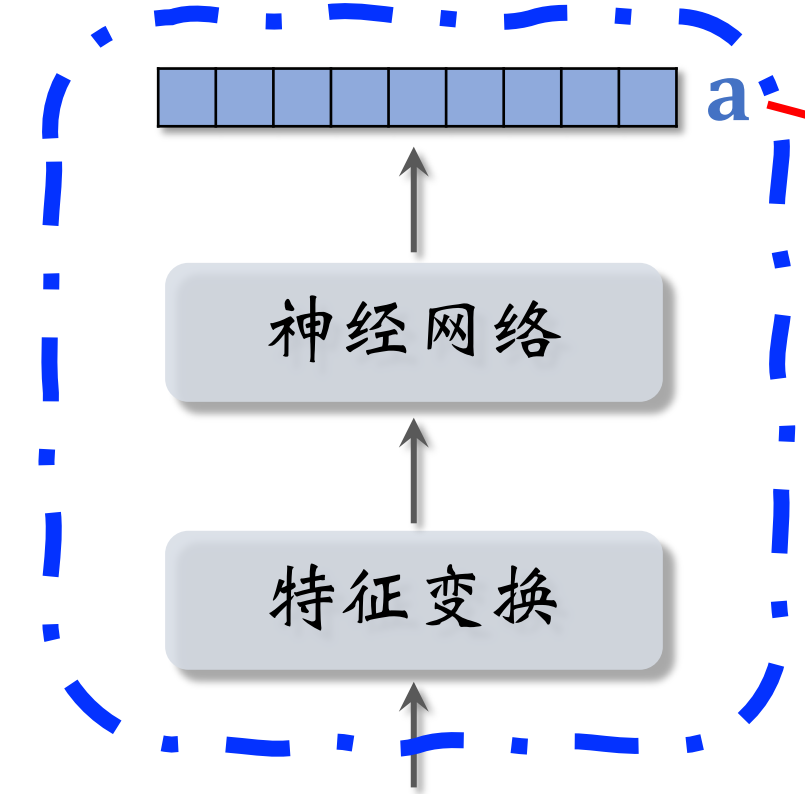
离线存储

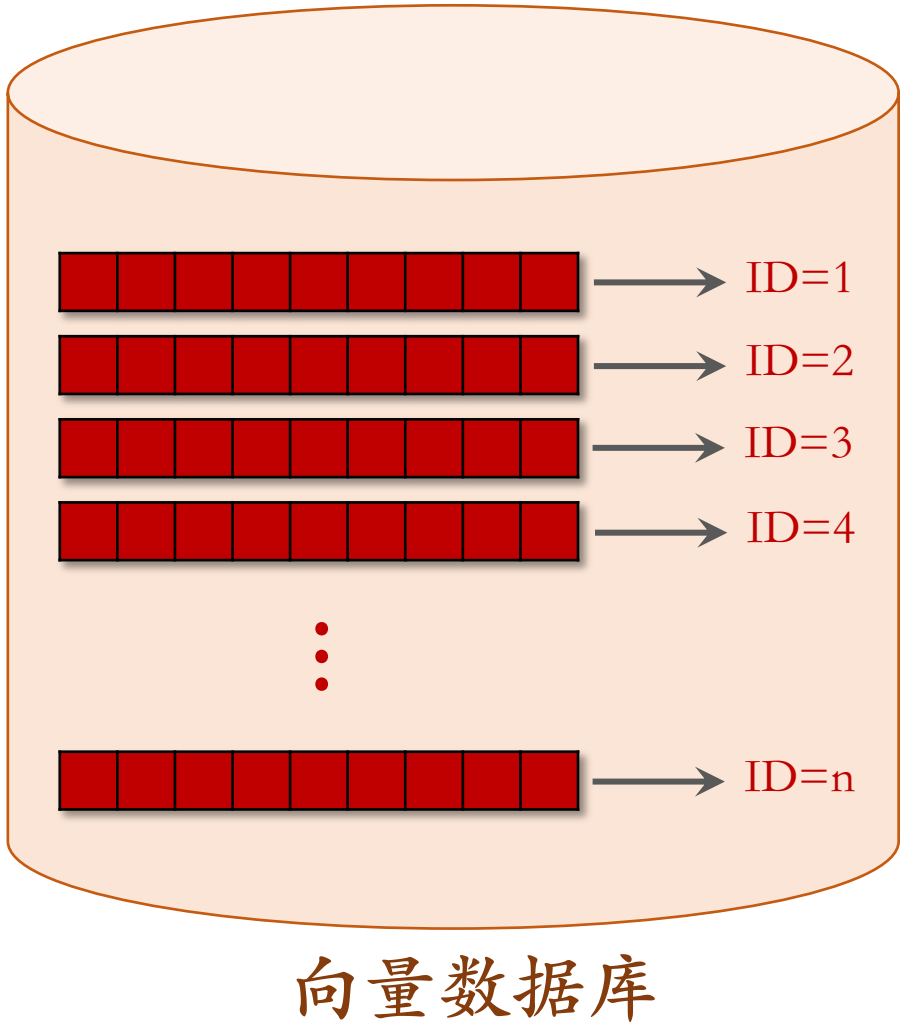




线上召回

给定用户ID和特征，
在线上计算向量 a。





双塔模型的召回

离线存储：把物品向量 $\mathbf{b}$ 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 $\mathbf{b}$ 。
2. 把几亿个物品向量 $\mathbf{b}$ 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。

双塔模型的召回

离线存储：把物品向量 **b** 存入向量数据库。

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。

双塔模型的召回

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。

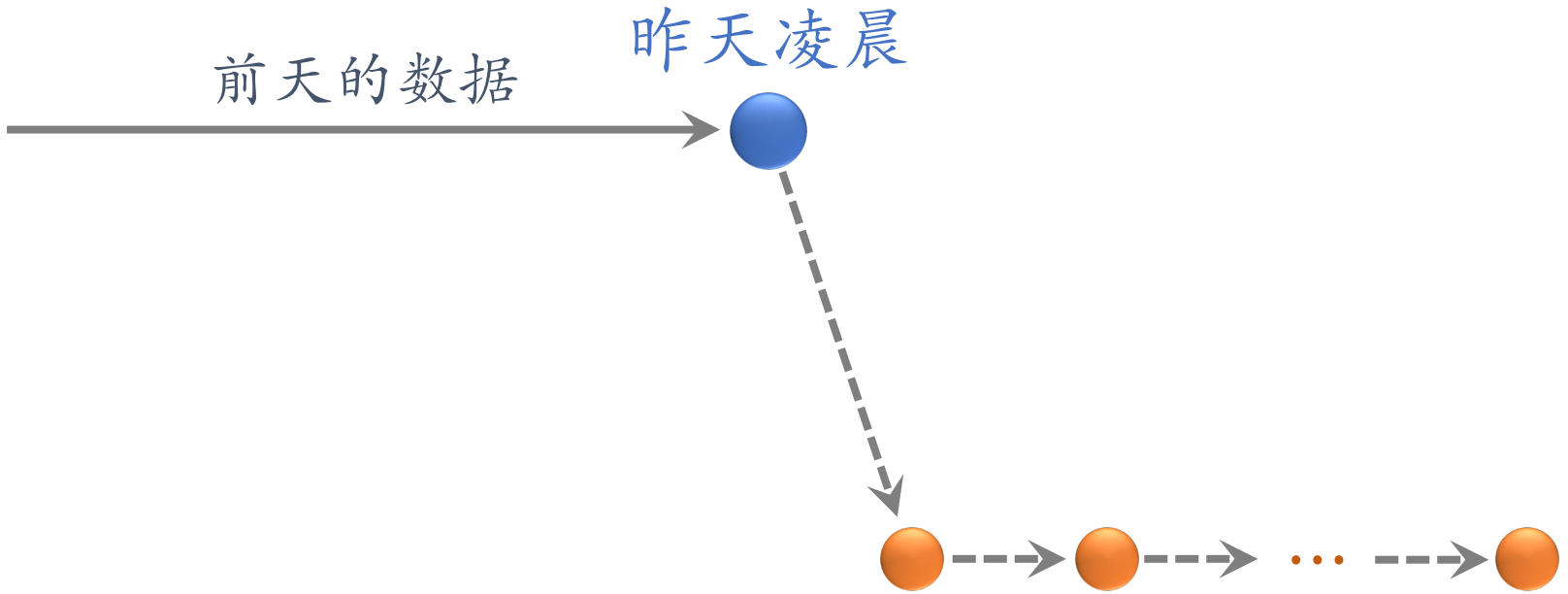
全量更新 vs 增量更新

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。（不更新神经网络其他部分的参数。）
- 发布用户 ID Embedding，供用户塔在线上计算用户向量。

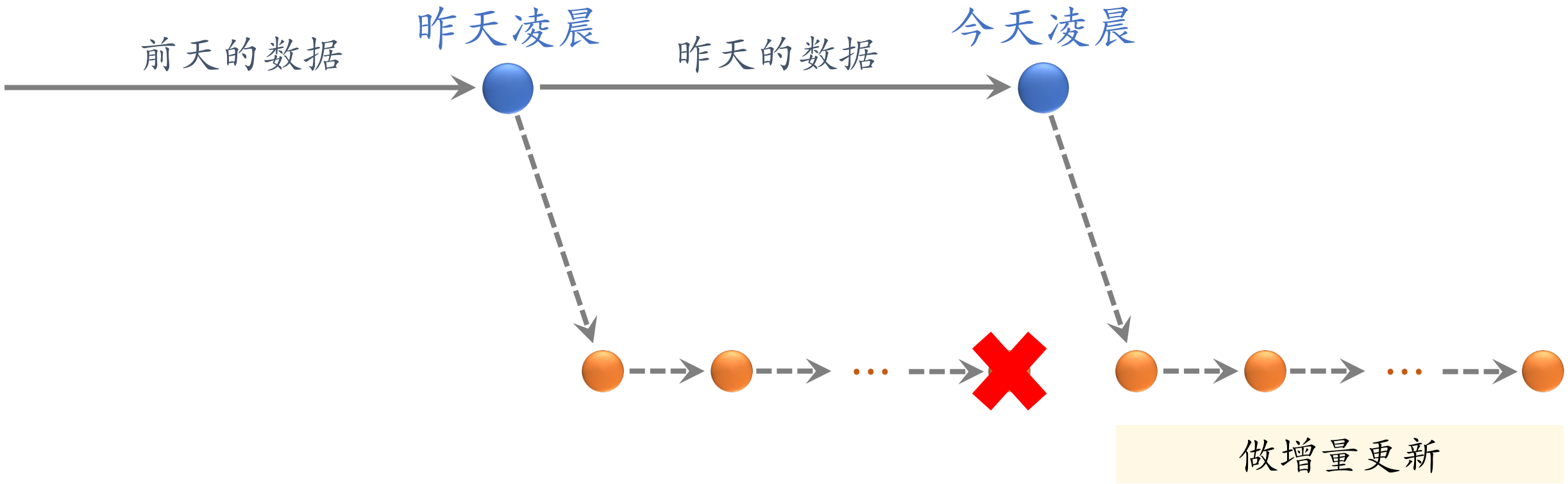
全量更新 vs 增量更新

基于前天的全量模型，用前天的数据，做全量更新。



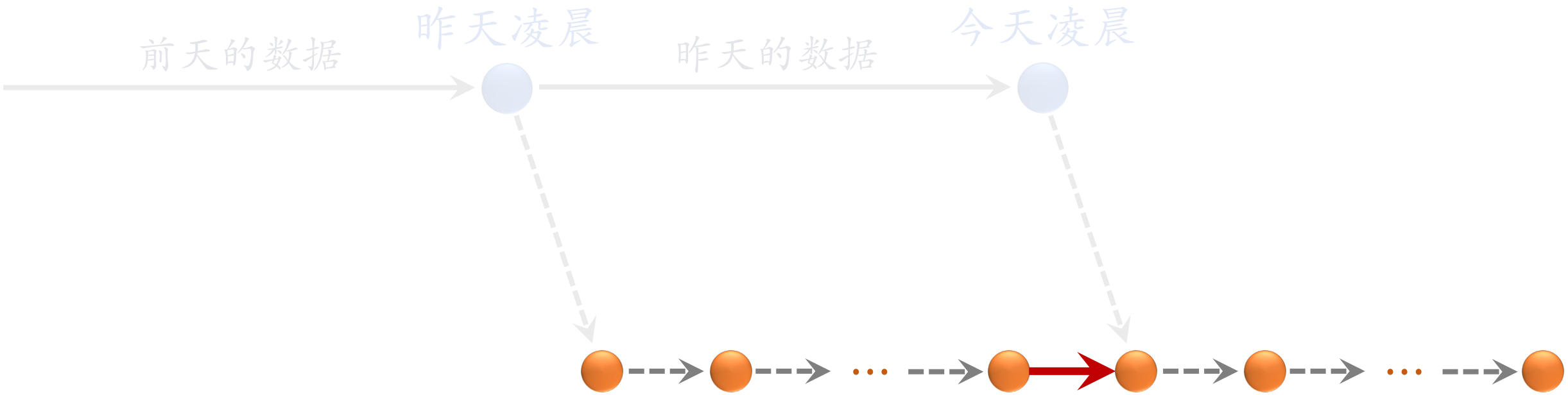
全量更新 vs 增量更新

基于昨天的全量模型，用昨天的数据，做全量更新。



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？



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？

- 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：全体物品（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 **a**。
- 把 **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。