南开大学

Nankai University

南　　开　　大　　学

计　算　机　学　院

大数据计算及应用期末实验报告

推荐系统算法及其优化报告

姓名：申健强 是忻言 仇科文

学号：2313119 2311848 2312237

专业：计算机科学与技术

指导教师：杨征路

2025 年 6 月 8 日

目录

一、 实验问题重述

在本项目中，我们需要基于 train.txt 中的用户-物品的交互数据数据对 Test.txt 文件中未知的用户–物品对 (u, i) 预测评分。

二、 数据集分析与预处理

(一) 数据集及其特征分析

在本次实验中，我们有两大主要的数据集：

- train.txt 训练集，其包含了用户与物品的交互打分数据
- test.txt 测试集，其中包含待预测打分的用户与物品编号

两个数据集的数据格式大致如下：

```
train.txt
<user id>|<numbers of rating items>
<item id>   <score>

test.txt
<user id>|<numbers of rating items>
<item id>
```

我们对数据进行了统计分析，得到如下表格

测试集统计	
训练集中未出现用户	12
用户: 测试集总数	610
用户: 未出现占比	1.97%
训练集中未出现物品	647
物品: 测试集总数	3618
物品: 未出现占比	17.88%

训练集统计	
总评分数	90854
用户数	598
物品数	9077
用户平均评分数	151.93
物品平均被评数	10.01

评分分布	
平均值	69.882119
标准差	20.780145
最小值	10
25% 分位数	60
中位数	70
75% 分位数	80
最大值	100

可以很明显的发现，在测试集中出现了很多训练集中没出现过的问题，这构成了推荐系统中的一个很经典的问题，冷启动问题，这个问题我们将在实验结果之后进行讨论。

(二) 数据预处理

在本次实验中，我们进行了不同的方法的尝试，对于不同的方法有不同的处理方法，以下我们分别进行大致的原理介绍。

1. **ID 映射** 原始用户和物品 ID 通常是非连续的字符串或大整数，不利于矩阵或张量运算。通过将每个 ID 映射为从 0 开始的连续整数索引，可方便地构造稠密矩阵与稀疏张量，并提升存储与计算效率。
2. **训练/验证集划分** 为了对模型泛化能力进行评估，将数据集按照一定比例（如 80:20）随机拆分为训练集和验证集。该拆分既能保证训练过程中的参数更新，又可以在训练后期用于监控过拟合与调参。

3. **评分标准化**

   - **Z-score 标准化**：将原始评分 x 转换为

   $$z = \frac{x - \mu}{\sigma},$$

   其中 μ 为评分均值，σ 为标准差。标准化后数据服从零均值、单位方差，有助于梯度下降等优化方法的稳定收敛。

4. **用户–物品矩阵构建** 构造矩阵 $R \in \mathbb{R}^{m \times n}$，其中 m 和 n 分别为用户数和物品数，R_{ui} 为用户 u 对物品 i 的评分，缺失值通常填为零或特殊标记，以适配不同矩阵分解与图模型。

5. **统计特征与中心化**

   - 计算用户平均评分

   $$\bar{r}_u = \frac{1}{|I_u|} \sum_{i \in I_u} R_{ui}.$$

   - 构造中心化矩阵 $\tilde{R}$，其中

   $$\tilde{R}_{ui} = R_{ui} - \bar{r}_u$$

   对缺失值忽略或补零，以消除不同用户评分尺度的偏差，这对于协同过滤等一系列模型来说是很重要的。

6. **相似度计算** 协同过滤方法中，常使用余弦相似度：

   $$\text{sim}(u, v) = \frac{\tilde{R}_u \cdot \tilde{R}_v}{|\tilde{R}_u|, |\tilde{R}_v|},$$

   用于衡量用户（或物品）向量间的角度相似性，指导基于邻域的推荐。

7. **批量化数据加载** 将预处理后的索引与评分打包为数值张量，以适配深度学习框架的批次化迭代。批量（batch）机制有助于并行计算与梯度估计的稳定性。

8. **隐式反馈集构建** 对于 SVD++ 模型，需要记录每个用户的交互集合

   $$N(u) = i :; R_{ui}; \text{存在},$$

   用于计算隐式反馈项，增强模型对未评分行为的理解。

9. **归一化邻接矩阵** 图卷积网络（GCN）中，将用户–物品交互构建为二分图邻接矩阵 A，再通过

   $$D_{ii} = \sum_j A_{ij}, \ \hat{A} \qquad\qquad = D^{-1/2} A D^{-1/2},$$

   实现对节点特征的双向传播与归一化，保证信息在多层网络中的稳定传递。

三、 算法原理

由上所述我们实现了若干推荐模型的探索，接下来我们将逐一介绍其原理及优劣，对于课上学习过的部分，我们不作赘述，我们讲述我们进行的拓展和尝试。

(一) NCF 模型

模型 在课堂上, 我们学习了传统协同过滤模型, 其交互函数仅为两个向量间的线性内积。Neural Collaborative Filtering (NCF) 将交互函数扩展为可学习的非线性映射，利用多层感知机 (MLP) 捕捉更复杂的用户–物品交互特征。

原理 令用户 u 的嵌入向量为 $\mathbf{e}_u \in \mathbb{R}^d$，物品 i 的嵌入向量为 $\mathbf{e}_i \in \mathbb{R}^d$，拼接后得到输入:

$$\mathbf{x} = \begin{bmatrix} \mathbf{e}_u \\ \mathbf{e}_i \end{bmatrix} \in \mathbb{R}^{2d}.$$

通过 L 层 MLP 学习非线性交互:

$$\mathbf{h}^{(1)} = \phi(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}), \quad \mathbf{h}^{(l)} = \phi(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}), \quad l = 2, \dots, L,$$

其中 $\phi(\cdot)$ 为激活函数（如 ReLU)。网络输出为:

$$\hat{r}_{ui} = \mathbf{w}_o^\top \mathbf{h}^{(L)} + b_o.$$

对应的带正则化均方误差损失:

$$\mathcal{L}_{\text{NCF}} = \frac{1}{|\mathcal{K}|} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \hat{r}_{ui})^2 + \sum_{l=1}^{L} \lambda_l \|\mathbf{W}^{(l)}\|_F^2 + \lambda_o \|\mathbf{w}_o\|_2^2.$$

(二) HybridCF 模型

模型架构 HybridCF 在 NCF 神经协同过滤的基础上，将经典协同过滤（矩阵分解中对用户和物品的低维嵌入）与神经网络（MLP）的非线性建模能力结合起来:

- **Embedding 层**: 为每个用户 u 和物品 i 学习 d 维嵌入向量 $\mathbf{e}_u, \mathbf{e}_i \in \mathbb{R}^d$，与矩阵分解中的隐向量对应。
- **拼接-MLP 头**: 将两个嵌入拼接成 $\mathbf{x} = [\mathbf{e}_u; \mathbf{e}_i] \in \mathbb{R}^{2d}$, 经过一层全连接 + ReLU + Dropout, 再由一维线性层输出预测评分。
- **杂交特性**: 通过保留 CF 的嵌入表达，实现了对稀疏评分矩阵的高效建模；通过 MLP 的非线性能力，捕捉高阶交互。

原理

$$\begin{aligned}
&\mathbf{e}_u = \text{Embed}_u(u), \quad \mathbf{e}_i = \text{Embed}_i(i), \\
&\mathbf{x} = [\mathbf{e}_u; \mathbf{e}_i], \\
&\mathbf{h}_1 = \text{ReLU}\big(\mathbf{W}_1\,\mathbf{x} + \mathbf{b}_1\big), \quad \mathbf{h}_1' = \text{Dropout}(\mathbf{h}_1), \\
&\hat{r}_{ui} = \mathbf{W}_2\,\mathbf{h}_1' + b_2.
\end{aligned}$$

损失函数同样采用带 L_2 正则化的均方误差:

$$\mathcal{L}_{\text{HybridCF}} = \frac{1}{|\mathcal{K}|} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \hat{r}_{ui})^2 + \lambda_1 \|\mathbf{W}_1\|_F^2 + \lambda_2 \|\mathbf{W}_2\|_2^2.$$

值得一提的是，这也是我们表现最好的模型。我们提交的源码和实验结果均是这种方法产出的结果。

(三) PMF 模型

在课堂上我们学习过基于矩阵分解推荐算法，在查阅资料后我们在这一方向进行了两个方向的尝试，分别是概率矩阵分解与 SVD++。

模型介绍 在经典矩阵分解（SVD）中，我们直接以

$$\hat{r}_{ui} = \mathbf{p}_u^\top \mathbf{q}_i$$

作为评分预测，但缺少对噪声与参数不确定性的刻画。概率矩阵分解（PMF）则从贝叶斯视角出发，引入概率分布，将观测模型与先验分布结合，用最大后验估计（MAP）同时完成参数估计与正则化。

为何引入概率分布

- **观测噪声建模**：假设真实评分 r_{ui} 在给定用户、物品潜在向量时，有高斯噪声干扰，

  $$p(r_{ui} \mid \mathbf{p}_u, \mathbf{q}_i, \sigma^2) = \mathcal{N}\big(r_{ui} \mid \mathbf{p}_u^\top \mathbf{q}_i,\ \sigma^2\big).$$

  这样可以更自然地解释数据的不确定性与噪声水平。

- **先验正则化**：对 $\mathbf{p}_u, \mathbf{q}_i$ 分别给出零均值高斯先验，

  $$p(\mathbf{p}_u \mid \sigma_p^2) = \mathcal{N}(\mathbf{0}, \sigma_p^2 \mathbf{I}), \quad p(\mathbf{q}_i \mid \sigma_q^2) = \mathcal{N}(\mathbf{0}, \sigma_q^2 \mathbf{I}),$$

  等价于在损失中自动加入 L_2 正则项，平衡数据拟合与模型复杂度。

- **MAP 推导**：最大化完整数据后验

  $$p(\{\mathbf{p}_u\}, \{\mathbf{q}_i\} \mid R) \propto \prod_{(u,i)\in\mathcal{K}} p(r_{ui} \mid \mathbf{p}_u, \mathbf{q}_i, \sigma^2) \prod_u p(\mathbf{p}_u \mid \sigma_p^2) \prod_i p(\mathbf{q}_i \mid \sigma_q^2)$$

  等价于最小化带权 MSE 加正则项的目标:

  $$\mathcal{L}_{\text{PMF}} = \frac{1}{2\sigma^2} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mathbf{p}_u^\top \mathbf{q}_i)^2 + \frac{1}{2\sigma_p^2} \sum_u \|\mathbf{p}_u\|^2 + \frac{1}{2\sigma_q^2} \sum_i \|\mathbf{q}_i\|^2.$$

(四) SVD++ 模型

模型介绍 经典 SVD（如 Funk-SVD 或 Biased MF）仅使用显式评分和偏置项:

$$\hat{r}_{ui}^{\text{SVD}} = \mu + b_u + b_i + \mathbf{p}_u^\top \mathbf{q}_i.$$

而 SVD++ 在此基础上，进一步考虑用户的**隐式反馈**（所有交互过的物品），以改进用户表征。

与 SVD 的不同

- **显式 vs. 隐式反馈**
  - SVD 只依赖 r_{ui} 明确打分，忽略用户的查看、点击、收藏等隐式行为。虽然在我们本次作业中这些特征并不存在，但其存在较好的实用意义，我们也对其进行了拓展。
  - SVD++ 将用户对过往所有物品 $j \in N(u)$ 的隐式交互也纳入模型，通过向量 $\mathbf{y}_j$ 聚合到用户表示中。

- **用户表示增强**

$$\mathbf{p}_u^{\text{SVD++}} = \mathbf{p}_u + \underbrace{\frac{1}{\sqrt{|N(u)|}} \sum_{j \in N(u)} \mathbf{y}_j}_{\text{隐式反馈聚合}}$$

这样每个用户的最终隐向量同时包含显式偏好和隐式行为信息，通常能显著提升稀疏情况下的预测准确性。

- **预测公式差异**

$$\hat{r}_{ui}^{\text{SVD++}} = \mu + b_u + b_i + \mathbf{q}_i{}^\top \Big(\mathbf{p}_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} \mathbf{y}_j \Big).$$

- **目标函数** SVD++ 的损失相比 SVD 多了一组隐式向量的正则化：

$$\mathcal{L}_{\text{SVD++}} = \sum_{(u,i) \in \mathcal{K}} (r_{ui} - \hat{r}_{ui}^{\text{SVD++}})^2 + \lambda \Big(b_u^2 + b_i^2 + \|\mathbf{p}_u\|^2 + \|\mathbf{q}_i\|^2 + \sum_{j \in N(u)} \|\mathbf{y}_j\|^2 \Big).$$

（五） LightGCN 模型

在相关课程的了解中，我们意识到深度学习和图卷积对于推荐系统而言是一种相当有效的方式，我们在进行实现时也尝试过相关的方向并进行改进（如下）。

动机与优势 推荐系统天然对应一个用户–物品二部图（bipartite graph），其中节点为用户或物品，边表示用户对物品的交互。Graph Convolution Network (GCN) 可以在图结构上进行消息传递，将高阶邻居信息融入节点表示。LightGCN 特别简化了 GCN：

- 去除非线性变换与特征变换（no feature transformation, no activation），仅保留邻居聚合部分，避免了参数过多和过平滑问题。
- 多跳（K 层）聚合实现高阶连接（用户–物品–用户…），自然捕捉协同过滤中“二阶”及以上相似度。
- 轻量无参数（只在嵌入层初始化参数），计算与存储开销低，易于大规模推荐场景。

原理 令用户数为 M，物品数为 N，初始嵌入：

$$\mathbf{E}^{(0)} = \begin{bmatrix} \mathbf{E}_u \\ \mathbf{E}_i \end{bmatrix} \in \mathbb{R}^{(M+N) \times d}.$$

构建归一化邻接矩阵

$$\tilde{\mathbf{A}} = \mathbf{D}^{-1/2} (\mathbf{A} + \mathbf{I}) \mathbf{D}^{-1/2},$$

其中 $\mathbf{A}$ 是二部图邻接（用户–物品边），$\mathbf{D}$ 为度矩阵。第 k 层嵌入更新为

$$\mathbf{E}^{(k+1)} = \tilde{\mathbf{A}}\, \mathbf{E}^{(k)}, \quad k = 0, 1, \ldots, K-1.$$

经过 K 层后，将所有层嵌入取均：

$$\mathbf{E} = \frac{1}{K+1} \sum_{k=0}^{K} \mathbf{E}^{(k)}.$$

拆分为用户与物品最终嵌入 $\mathbf{E}'_u \in \mathbb{R}^{M \times d}$, $\mathbf{E}'_i \in \mathbb{R}^{N \times d}$，并以内积进行评分预测：

$$\hat{r}_{ui} = \mathbf{e}'_u{}^\top \mathbf{e}'_i.$$

训练目标为最小化

$$\mathcal{L}_{\text{LightGCN}} = \frac{1}{|\mathcal{K}|} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \hat{r}_{ui})^2.$$

（六） LightGCNReg 模型

改进动机 虽然 LightGCN 能有效捕捉多跳连接的高阶协同信号，但纯内积预测忽略了整体偏置效应，我们发现其推荐效果并不好。LightGCNReg 在保留其轻量图卷积骨架的同时，引入回归头，从而：

- 建模全局均值与用户/物品偏置，矫正打分偏移；
- 保留图卷积嵌入的高阶聚合特性；
- 仍保持参数量低、计算高效。

数学形式 利用 LightGCN 得到的最终嵌入 $\mathbf{e}'_u, \mathbf{e}'_i$，预测函数扩展为

$$\hat{r}_{ui} = \mu + b_u + b_i + {\mathbf{e}'_u}^\top \mathbf{e}'_i,$$

其中

$$\mu = \frac{1}{|\mathcal{K}|} \sum_{(u,i)\in\mathcal{K}} r_{ui}, \quad b_u, b_i \text{ 为可学习的用户/物品偏置。}$$

损失函数在 MSE 基础上加入所有参数的 L_2 正则：

$$\mathcal{L}_{\text{LightGCNReg}} = \frac{1}{|\mathcal{K}|} \sum_{(u,i)\in\mathcal{K}} \left(r_{ui} - \hat{r}_{ui}\right)^2 + \lambda_\mu \mu^2 + \lambda_u \sum_u b_u^2 + \lambda_i \sum_i b_i^2 + \lambda_e \left\|\mathbf{E}\right\|_F^2.$$

（七） 自注意力自动编码器模型 (atten_autoenc)

模型 在我们完成上面的尝试后适逢人工智能导论课程介绍了 transformer 架构，于是我们也进行了相关尝试，结合自动编码器与自注意力机制，学习物品表示并进行重构。

原理 编码器：

$$\mathbf{h}_1 = \text{ReLU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1), \quad \mathbf{e} = \text{LayerNorm}(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2).$$

自注意力：通过 L 层 Transformer Encoder：

$$\mathbf{e}' = \text{TransformerEncoder}(\mathbf{e}).$$

解码器：

$$\mathbf{h}_2 = \text{ReLU}(\mathbf{W}_3\mathbf{e}' + \mathbf{b}_3), \quad \hat{\mathbf{x}} = \mathbf{W}_4\mathbf{h}_2 + \mathbf{b}_4.$$

重构损失：

$$\mathcal{L} = \|\mathbf{x} - \hat{\mathbf{x}}\|^2.$$

四、 实现代码分析

在我们的实现中 HybridCF 模型表现最优，在此我们介绍这个模型的实现。[1]

[1]注：其余模型的实现可以查看仓库https://github.com/sjq0098/Data_Mining.git

(一) 数据加载与索引映射

首先,我们从原始文本文件中读取训练集和测试集数据。训练集文件中每一行为“用户 ID|”(表示用户切换）或“物品 ID 评分”格式；测试集文件每一行为“用户 ID|”或单纯的“物品 ID”。读取完成后，得到如下两个数据框：

$$\texttt{train\_df} = \{\texttt{user}, \texttt{item}, \texttt{score}\}, \quad \texttt{test\_df} = \{\texttt{user}, \texttt{item}\}.$$

随后，将全体出现过的用户和物品分别收集并建立索引映射：

$$\texttt{user2idx} : u \mapsto [0, |\mathcal{U}| - 1], \quad \texttt{item2idx} : i \mapsto [0, |\mathcal{I}| - 1].$$

对训练集和测试集增加两列 `u_idx`, `i_idx`，映射失败者（即测试集中新出现的用户/物品）用 -1 填充并转换为整型，方便后续处理。

(二) 数据集与数据加载器

此处我们定义一个继承自 `torch.utils.data.Dataset` 的 `RatingDataset` 类：

- 构造函数根据传入的 DataFrame 构造三条张量：用户索引 **u**、物品索引 **i**，以及评分 **r**（若有评分标签)。
- `__len__` 返回样本数量，`__getitem__` 返回单条样本（若无评分，则仅返回 (u, i) 对)。

利用 `DataLoader` 将整个训练集打包为批量（batch_size=512)，并在每个 epoch 进行随机打乱。

(三) HybridCF 模型结构

模型整体结构如前文中我们已经进行了简单的介绍, 具体定义如下：

```
class HybridCF(nn.Module):
    def __init__(self, num_users, num_items, embed_dim, hidden_dim, dropout):
        super().__init__()
        self.user_emb = nn.Embedding(num_users, embed_dim)
        self.item_emb = nn.Embedding(num_items, embed_dim)
        nn.init.xavier_uniform_(self.user_emb.weight)
        nn.init.xavier_uniform_(self.item_emb.weight)
        self.fc1 = nn.Linear(embed_dim * 2, hidden_dim)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(dropout)
        self.fc2 = nn.Linear(hidden_dim, 1)
```

前向传播（Forward）过程：给定用户索引向量 **u** 和物品索引向量 **i**，先分别获取其嵌入向量

$$\mathbf{e}_u = \text{Embedding}_u(\mathbf{u}), \quad \mathbf{e}_i = \text{Embedding}_i(\mathbf{i}),$$

再在 $\dim = 1$ 方向上拼接 $\mathbf{x} = [\mathbf{e}_u; \mathbf{e}_i] \in \mathbb{R}^{2d}$，通过全连接层、激活、Dropout 后映射到评分预测：

$$\begin{aligned} \mathbf{h} &= \text{ReLU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1), \\ \mathbf{h}' &= \text{Dropout}(\mathbf{h}), \\ \hat{r} &= \mathbf{W}_2\mathbf{h}' + b_2. \end{aligned}$$

（四） 模型初始化与优化器设置

模型参数在 CPU 上初始化，超参数选择如下：

- 嵌入维度 embed_dim = 16（网格搜索最佳）。
- 隐藏层维度 hidden_dim = 128（网格搜索最佳）。
- Dropout 比例 dropout = 0.2（网格搜索最佳）。
- 损失函数：均方误差损失 $\mathcal{L} = \frac{1}{N}\sum(\hat{r} - r)^2$。
- 优化器：Adam，学习率 $\alpha = 10^{-3}$。

（五） 全量训练过程

设置训练轮次 $n_{\text{epochs}} = 20$。每一轮中：

1. 切换模型到训练模式 `model.train()`。
2. 遍历每个批次 $(\mathbf{u}, \mathbf{i}, \mathbf{r})$，在前向和反向传播后更新参数：

$$\begin{aligned}\hat{\mathbf{r}} &= \texttt{model}(\mathbf{u}, \mathbf{i}),\\ \mathcal{L} &= \frac{1}{B}\sum(\hat{r}_j - r_j)^2,\\ \theta &\leftarrow \theta - \alpha\,\nabla_\theta \mathcal{L}.\end{aligned}$$

3. 累积并打印本轮平均训练损失及对应的 RMSE。

最后，将模型参数保存为 `hybrid_cf_final.pth`。

（六） 测试集预测与结果保存

测试阶段：

- 构造仅含 (u, i) 对的 `RatingDataset`，并用 DataLoader 按批次推断。
- 对于映射失败的样本（$u = -1$ 或 $i = -1$），我们先在批次内部用零占位，预测后再统一替换为训练集全局评分均值 $\bar{r}$。
- 将最终预测结果按用户分组写入文本文件，每组首行为“用户 ID| 该用户预测条目数”，随后逐行写入“物品 ID 预测评分”。并额外导出为 CSV 以便后续分析。

至此我们介绍完了我们的最佳表现的模型实现，接下来我们将对实验结果进行分析。

五、 实验结果及分析

（一） 实验结果及分析

我们将上述的几种模型在数据集上进行训练与测试，得到了基于验证集的 RMSE 对比。

在上述模型的训练中我们都没有刻意进行参数优化，我们可以很明显的看到，混合协同过滤模型取得了很好的效果，RMSE 达到 16.9381，以下我们将进行详细分析。

表 1: 各模型验证集 RMSE 对比

模型	MF	LightGCN	LightGCNReg	Atten	UserCF	**HybridCF**	PMF	SVD++	NCF
Val RMSE	17.4658	40.1701	19.3677	24.1709	18.3493	**16.9381**	21.8715	18.1006	18.3493

不同模型类别之间的对比 整体来看，我们提出的 **HybridCF 模型**在所有模型中表现最佳，验证集 RMSE 达到 **16.9381**，显著优于其他方法，体现了该模型在结合线性嵌入表示与非线性 MLP 表达能力方面的优势。

传统基于矩阵分解的模型（如 MF、PMF、SVD++）RMSE 分别为 17.4658、21.8715 和 18.1006，表现较为稳定。其中，**SVD++** 因为在数据集中并不存在相关数据，所以其表现和普通矩阵分解并无太大差别甚至略微落后普通矩阵方法。

图神经网络类的 **LightGCN** 表现最差（RMSE 高达 40.1701），这其实比较出乎我们的预料，后来我们分析后得知原因在于其评分预测完全依赖嵌入间的内积，忽略了评分偏置与尺度控制。改进后的 **LightGCNReg** 加入了回归头，有效缓解了上述问题，RMSE 降至 19.3677，说明偏置建模对评分预测的重要性。

深度神经网络类的 **NCF** 模型相比 MF 略有提升（18.3493 vs 17.4658），但并不显著，说明在当前数据集规模下，简单 MLP 的非线性拟合能力尚未完全发挥，可能受到过拟合与训练难度影响。

自注意力机制模型 **Atten** 表现较差（RMSE 24.1709），原因可能包括网络结构复杂、训练样本不足，以及注意力结构对评分重构任务不够契合，说明直接迁移 Transformer 架构到评分预测任务上存在一定困难。

浅层模型 vs 深层模型 - UserCF 模型基于显式用户相似度建模，尽管无参数但可解释性强，表现也较好（18.3493）。- Atten 模型结构复杂但效果反而更差，提示我们：**复杂模型未必适合稀疏评分预测任务**，尤其是在缺少丰富上下文或交互行为的情形下，深层结构反而易过拟合。

(二) 调整表现最好模型的参数

我们对 HybridCF 模型进行了进一步的参数调整，通过网格搜索的方式搜索表现最好的参数组合，我们设置搜索网格如下:

```
    param_grid = {
    'embed_dim': [16, 32, 64],
    'hidden_dim': [32, 64, 128],
    'dropout': [0.1, 0.2],
    'lr': [1e-3, 5e-4],
    'batch_size': [512, 1024]
}
```

优化得到，表现最佳的参数组合是:

```
    embed_dim = 16,
    hidden_dim= 128,
    dropout   = 0.2
    lr=0.001,
    batch_size=512.
```

这样我们的模型表现进一步优化，来到 **16.4391**。其训练曲线如图1。

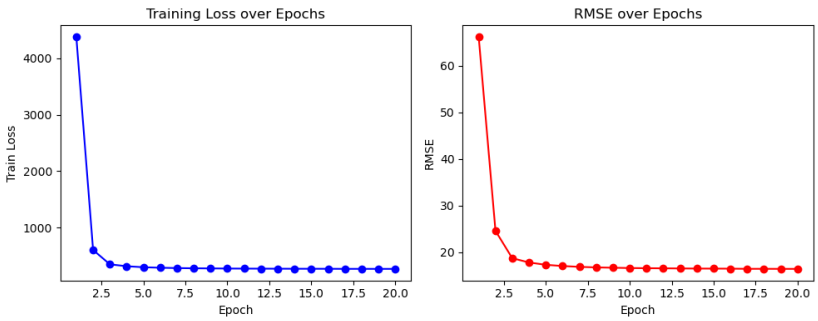

图 1: 训练过程可视化

六、 Discussion

结合上述数据特征与实验结果，我们重点讨论以下两点：

1. 冷启动问题 如前所述，测试集中存在 $\approx 2\%$ 的新用户和 $\approx 18\%$ 的新物品。对于这些冷启动实体，我们实现的模型（如 MF、PMF、LightGCN）基本无法直接给出有效预测，只能依赖全局均值或相似用户/物品的偏置进行粗略估计，导致误差显著上升。

如果存在其他交互信息，之前尝试的 SVD++ 所隐含的思想可能会更加有效, 我们不止查看评分信息，也关注用户与物品之间的交互，这样就可以关注用户与物品之间的隐藏关系乃至用户与用户之间的隐藏关系来助力冷启动问题的缓解。

2. 深度学习模型与传统模型的表现差异 尽管深度神经网络（如 NCF、自注意力编码器）在表达能力上更为灵活，但在本项目所用的稀疏评分数据集上表现并不突出，甚至弱于经典的矩阵分解模型（MF、PMF）和 HybridCF。主要原因包括：

- **样本规模不足**：深层网络需要大量多样化数据以避免过拟合，本实验评分样本仅约 9 万条，且上下文特征匮乏，限制了其非线性表达能力的发挥。
- **结构复杂度高**：Transformer 类结构（Atten_autoenc）虽然具有强大的建模能力，但也容易在稀疏环境下学习到无意义的注意力权重，从而拖累整体性能。
- **偏置与尺度控制**：LightGCN 完全依赖内积计算，而忽略用户/物品偏置，导致评分尺度失真；而传统模型（MF、SVD++）及 HybridCF 中均显式或隐式地进行了偏置建模，能更好地校正用户评分习惯差异。

因此，在本项目中，我们观察到：**适度结合传统嵌入与非线性层（如 HybridCF）的“浅层 + 非线性”混合方法，往往在稀疏现实场景中优于纯深度模型或传统模型本身**。

七、 总结

在本项目中, 我们基于 `train.txt` 中的用户-物品交互数据, 针对 `test.txt` 中未知的用户-物品对 (u, i) 进行了评分预测研究。通过对多种推荐模型的实现与比较, 实验结果表明, **HybridCF** 模型在验证集上取得了最佳表现，其 RMSE 为 **16.9381**，显著优于其他模型。这一成功归因于 HybridCF 结合了传统协同过滤的嵌入表示与神经网络的非线性建模能力，有效捕捉了稀疏数据环境下的复杂用户-物品交互特征。

进一步分析，传统矩阵分解模型（如 MF、PMF）表现稳定但受限于线性表达能力，RMSE 分别为 17.4658 和 21.8715；图神经网络模型（如 LightGCN）因忽略偏置建模而表现最差（RMSE 40.1701）；深度神经网络模型（如 NCF、Atten_autoenc）虽具有较强表达潜力，但因数据稀疏与结构复杂性未达预期效果，RMSE 分别为 18.3493 和 24.1709。相比之下，HybridCF 通过“浅层 + 非线性”的混合设计，在稀疏场景中取得了平衡与高效。

此外，我们对 HybridCF 进行了参数调优，通过网格搜索优化超参数，最终将 RMSE 进一步提升至 **16.4391**。

总之，在本次任务中，我们对推荐系统算法，进行了不同方向的尝试，基于课内的方法进行了一些延申，最终较好的完成了本次实验。

致谢：在此本学期本课程的实验部分就结束了，在此我们特别感谢杨老师耐心细致的讲解，老师深入浅出的分析让我们对推荐算法有了清晰的认知，并对数据挖掘这个领域产生了浓厚的兴趣，并也了解了很多本领域的知识，受益颇深。