AutoDenoise：面向推荐的自动数据实例去噪

Weilin Lin 香港城市大学 mlmr.lin@gmail.com

Xiangyu Zhao* 香港城市大学 xianzhao@cityu.edu.hk

Yejing Wang 香港城市大学 yejing.wang@my.cityu.edu.hk

Yuanshao Zhu 香港城市大学 南方科技大学 huys2019@mail.sustech.edu.cn

Wanyu Wang 香港城市大学 anyuwang4-c@my.cityu.edu.hk

arXiv:2303.06611v1 [cs.IR] 12 Mar 2023

摘要

历史用户-物品交互数据集对于训练现代推荐系统以预测用户偏好至关重要。然而，在大多数推荐场景中，用户行为具有任意性，导致记录了大量含噪数据实例，无法充分反映用户的真实兴趣。尽管推荐系统领域涌现出大量去噪研究，但这些研究都受到数据分布高度动态变化的影响。本文提出一种基于深度强化学习（DRL）的框架 AutoDenoise，并设计了实例去噪策略网络，以实例选择的方式对深度推荐系统中的数据实例进行去噪。具体而言，AutoDenoise 在深度强化学习中充当智能体，自适应地选择无噪声且具有预测性的数据实例，随后可将这些实例直接用于训练具有代表性的推荐模型。此外，我们设计了一种交替式两阶段优化策略，以适当地训练和验证 AutoDenoise。在搜索阶段，我们旨在训练具备实例去噪能力的策略网络；在验证阶段，我们找出并评估训练后的策略网络所选出的去噪数据实例子集，从而验证其去噪能力。我们通过大量实验，验证了 AutoDenoise 与多种具有代表性的推荐系统模型结合使用时的有效性。

CCS概念

• **信息系统** →**推荐系统**。

关键词

实例去噪、推荐系统、强化学习

*Xiangyu Zhao 是通讯作者。

允许免费为个人或课堂使用而制作本作品的全部或部分数字或纸质副本，前提是这些副本不得用于营利或商业利益，也不得为此目的进行分发，并且副本必须保留本声明和首页上的完整引文。本作品中归其他作者所有的组件，其版权必须得到尊重。注明出处后，允许摘录。若要以其他方式复制、再版、发布到服务器或分发至列表，则须事先获得特定许可和/或支付费用。请向 permissions@acm.org 申请许可。
WWW '23，*2023*年*4*月*30*日至*5*月*4*日，美国得克萨斯州奥斯汀© 2023年版权归所有者/作者所有。出版权已授权给 ACM。ACM ISBN 978-1-4503-9416-1/23/04...$15.00 https://doi.org/10.1145/3543507.3583339

ACM 引用格式：
Weilin Lin、Xiangyu Zhao、Yejing Wang、Yuanshao Zhu 和 Wanyu Wang。2023年。AutoDenoise：面向推荐的自动数据实例去噪。载于《*2023*年*ACM Web*会议论文集》（*WWW '23*），*2023*年*4*月*30*日至*5*月*4*日，美国得克萨斯州奥斯汀。ACM，美国纽约州纽约市，9页。https://doi.org/10.1145/3543507.3583339

1 引言

推荐系统（RS）是信息爆炸时代的一项关键技术，能够为用户带来良好的体验，并为企业创造可观的经济效益 [8,13, 16]。作为一种数据驱动技术，推荐系统模型基于先前的数据实例（即交互日志）建模用户偏好 [25],，从而显著提升在线服务平台（如电子商务网站和流媒体视频应用）的服务质量 [5, 27]。一般而言，推荐系统基于历史用户-物品交互中记录的数据实例进行建模。然而，通过上述方法记录用户行为，难免会给推荐系统建模带来挑战。首先，现有的推荐数据集会不加区分地收集所有历史数据实例，其中一些可能是无法反映用户真实意图的噪声实例 [12]。例如，在点击率（CTR）预测中 [3, 9, 17, 28, 31,38, 40, 45], ，某些点击行为可能源于用户的好奇或误操作。此外，噪声实例与无噪声实例的特征几乎相同 [14],，这使得人类专家难以从记录数据中将它们区分开来。

为应对上述挑战，训练数据去噪正日益受到研究人员的关注。近期研究表明，在推荐系统中采用去噪技术，可以在计算成本相近的情况下，以更高的效率训练模型，并取得更好的性能 [7, 11, 32, 37]。通常，去噪方法包含“搜索”过程，用于找出噪声，以及“决策”过程，用于执行去噪动作。根据去噪方法的不同，现有工作大致可分为两类：基于选择的方法和基于重加权的方法。对于基于选择的方法 [6, 35], ，其目标是训练一个选择网络，以剔除噪声实例，从而为模型提供更具代表性的数据。对于基于重加权的方法 [29, 30],，其倾向于在模型训练过程中为噪声实例分配较低权重，以降低其影响。实践中，这些方法将损失值视为区分噪声与无噪声实例的指标，即训练期间损失值较高意味着数据实例含有噪声。

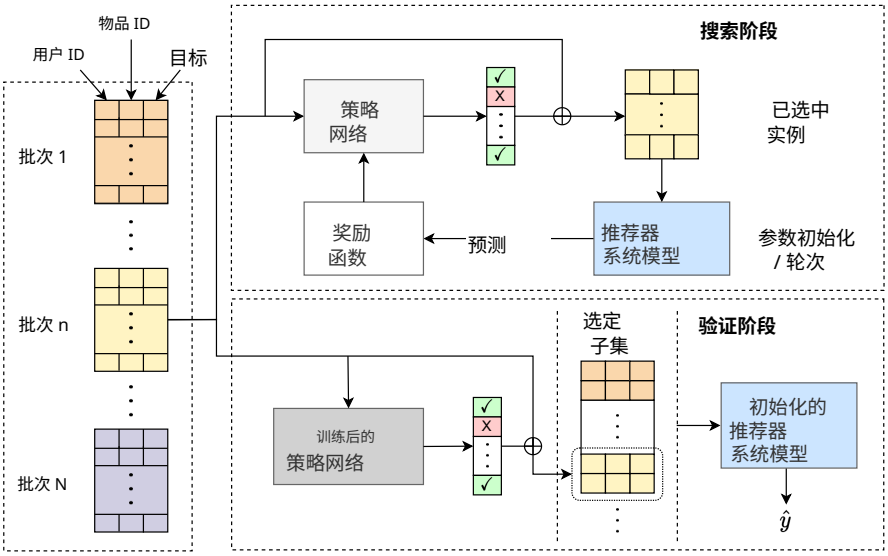

图1：所提框架概览。

然而，上述方法存在以下局限。首先，基于选择的方法严重依赖采样分布来进行决策 [36]，而当代在线环境中复杂且动态的数据分布往往会限制其性能。换言之，环境越动态，选择就越可能出现偏差。其次，影响模型训练过程的基于重加权的方法缺乏可迁移性。对于给定的模型或推荐任务，现有基于重加权的方法需要特定配置，这既耗时，也难以迁移到其他设置。因此，一个有前景的实例去噪方案应当具备鲁棒性，并易于迁移。

为弥补这一研究空白，我们提出了一种基于深度强化学习（DRL）的实例去噪框架，该框架能够对真实世界数据集的各种复杂数据分布进行建模，并筛选出无噪声子集。提出这一框架主要基于三个动机：（i）实践证明，强化学习方法在最优搜索问题中效果显著 [18, 19, 39, 41–44], ，因此也有望有效地区分噪声数据实例。（ii）深度强化学习中的策略网络用于从一组小批次中选择实例，使其能够从各种数据分布中捕获细粒度模式。这种设计恰好有助于缓解基于选择的方法所面临的选择偏差问题。（iii）分别训练用于去噪和预测过程的预测模型与策略网络，使去噪策略网络生成的无噪声数据子集具备天然的可迁移性。

然而，将基于深度强化学习的方法应用于实例去噪时，会面临以下挑战。一方面，不同小批次之间的数据分布差异显著，因此难以设计合适的奖励函数来优化策略网络。此外，在这种设置下，策略网络和推荐系统模型需要同时优化，因此难以用同一个数据批次恰当地训练二者。为应对上述挑战，我们提出了一种基于深度强化学习的实例去噪框架 AutoDenoise，并配备实例去噪策略网络。AutoDenoise 使用策略网络为小批次中的每个实例计算对应于“选择/不选择”动作的两个概率，并使用推荐系统模型评估被采样并“选择”的实例的性能。同时，我们基于推荐系统模型的结果设计了一种实例级奖励函数。具体而言，该奖励函数将数据实例当前的损失与其在之前搜索轮次中的损失进行比较，从而学习识别不同分布中的噪声数据实例。此外，我们提出了一种交替进行的两阶段优化策略，即搜索阶段和验证阶段，以恰当地训练和验证策略网络。随后，我们从头开始，在选定的数据子集上训练一个随机初始化的推荐系统模型，以评估新数据子集的有效性。本文的主要贡献总结如下：(i) 我们提出了一种基于深度强化学习的实例去噪框架，可自适应地过滤每个输入小批次数据实例中的噪声。据我们所知，这是首项针对点击率预测进行实例去噪的开创性工作；(ii) 我们设计了一种新颖的两阶段训练过程，以有效训练和验证策略网络，并生成可迁移的无噪声数据子集；(iii) 我们在三个公开基准数据集上验证了 AutoDenoise 的有效性，并证明了去噪数据集的可迁移性。

2 框架

本节将介绍 AutoDenoise 框架的概述、提出的两个阶段以及优化方法。

2.1 概述

如图1所示，整个框架包含两个运行阶段，即搜索阶段和验证阶段。

搜索阶段。在此阶段，采用深度强化学习，通过对模型和策略网络进行序贯优化，自动选择实例。具体而言，策略网络充当

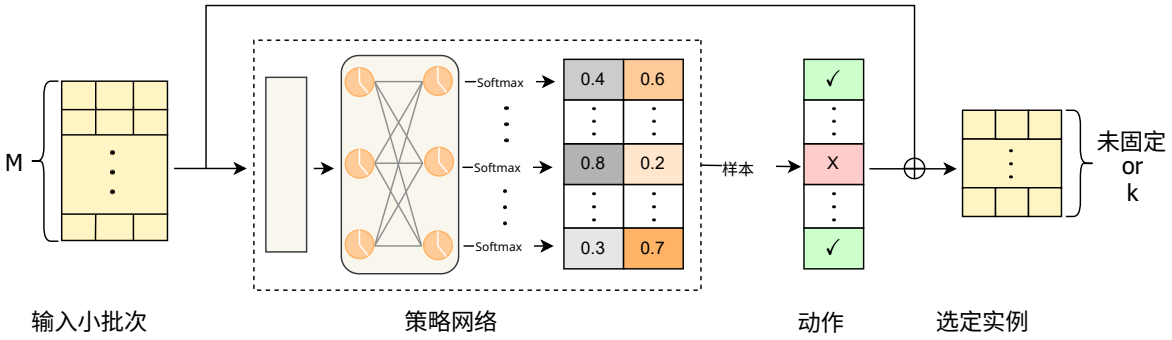

图 2：策略网络的选择过程。

深度强化学习智能体，以相应的概率执行“选择”和“取消选择”动作，从而移除噪声实例。随后，将选出的无噪声实例输入推荐系统模型，并根据推荐系统模型先前和当前的损失计算奖励，用于优化策略网络。

验证阶段。 与搜索阶段不同，此阶段旨在评估策略网络的性能。具体而言，在每轮搜索阶段结束后，也就是对整个训练集进行一次迭代后，流程便切换到验证阶段。在验证阶段，策略网络按批次预测整个训练集，并选取“选择”动作概率最高的前-k 个实例，这些实例可视为无噪声数据。最后，使用无噪声子集训练一个初始化的推荐系统模型，直至收敛，并将最终测试性能作为策略网络去噪能力的体现。

2.2 搜索阶段

本节重点介绍策略网络和深度强化学习构建中的核心要素（例如环境、状态、奖励等）。作为应用深度强化学习的实例去噪任务，我们将每个小批次视为输入策略网络的状态，然后根据输出概率执行深度强化学习动作（“选择”或“取消选择”）。深度强化学习通过将选定实例输入推荐系统模型与环境交互，并计算相应的奖励，以最大化策略网络的奖励。接下来，我们将深入介绍这些组成部分的细节。

环境。 RS 在深度强化学习中充当环境，接收一个由选定实例构成的小批次，并输出预测。为了公平地评估和优化策略网络，我们在每个训练轮次开始时初始化其参数，使输出预测能够与先前的预测进行比较。

状态。 状态是输入小批次，由一组数据实例组成。假设一个小批次有 M 个实例，总共有 N 个批次，整个训练集为 $\boldsymbol{X} = \{\boldsymbol{X}_1^n, ..., \boldsymbol{X}_M^n\}_{n=1}^N$，其中 $\boldsymbol{X}_m^n$ 表示训练轮次中第 n^{th} 个小批次的第 m^{th} 个数据实例。为了比较每个训练轮次中实例级别的预测结果，我们固定它们的顺序。换句话说，所有 $\boldsymbol{X}_m^n$ 在整个训练过程中都处于相同的位置。由于状态是输入小批次，因此 n^{th} 小批次可以用一般形式表示，即$s_n = [\boldsymbol{X}_1^n \cdots \boldsymbol{X}_m^n \cdots \boldsymbol{X}_M^n]$。策略网络的作用是为 s_n 中的每个实例打分，计算其概率，并确定“选择/不选择”动作。

策略与动作。 在去噪过程中，策略与动作分别可视为对实例进行“搜索”和“决策”的行为。因此，我们提出一种策略网络，根据状态 s_n 搜索无噪声实例，并对小批次 s_n中的每个实例执行动作 a_m^n。整个选择过程如图 2 所示。具体而言，给定一个数据批次 $s_n = [\boldsymbol{X}_1^n \cdots \boldsymbol{X}_m^n \cdots \boldsymbol{X}_M^n]$，策略网络首先通过嵌入查找操作将每个实例 $\boldsymbol{X}_m^n$ 转换为一个稠密向量 $\boldsymbol{E}_m^n$，

$$\boldsymbol{E}_m^n = \mathcal{A}\boldsymbol{X}_m^n \tag{1}$$

其中，$\mathcal{A}$ 由所有特征字段对应的可学习权重矩阵组成[1]。 $\boldsymbol{E}_m^n$ 表示与 $\boldsymbol{X}_m^n$对应的拼接特征嵌入向量。随后，将 $\boldsymbol{E}_m^n$ 输入带有非线性激活的多层感知机（MLP），以获得细粒度的特征表示 [23]。假设 MLP 总共有 $\{L+1\}$ 层，则上述对 l^{th} （$l \in [1, L]$）层的操作可表示为：

$$\boldsymbol{h}_l = \sigma\left(\boldsymbol{W}_l \boldsymbol{h}_{l-1} + \boldsymbol{b}_l\right) \tag{2}$$

其中， $\boldsymbol{h}_l$ 是 l^{th} 层的输出。 $\boldsymbol{W}_l$ 和 $\boldsymbol{b}_l$ 分别是对应层的可学习权重矩阵和偏置向量， $\sigma(\cdot)$表示非线性激活函数。通常，我们将 $ReLu$ 用作隐藏层的激活函数。然后，对输出层应用 $Softmax$ 函数，以获取每个实例执行“选择”和“取消选择”动作的概率。在这种设置下，动作 a_m^n 可以根据输出概率进行采样。需要提醒的是， $\boldsymbol{h}_0 = \boldsymbol{E}_m^n$ 表示第一个全连接层，而最后一层 $\boldsymbol{h}_L$ 是输出层。

奖励。 为了以期望的方式优化策略网络，即学习区分噪声实例，奖励设计应与噪声实例识别模型的性能相关联。基于这一思路，我们根据推荐系统模型的预测误差定义奖励，即比较预测结果与对应真实标签的损失值 L。为了避免数据实例之间的优化冲突，我们在训练搜索阶段分别考虑每个实例。具体而言，我们创建一个矩阵 $\boldsymbol{LMat} \in \mathbb{R}^{C \times MN}$，用于存储每个实例在先前和当前搜索轮次中的损失值，其中 C 和 MN 分别表示存储的轮次数量和损失值的数量。给定当前轮次的损失 L_{mn} （ m^{th} 实例

[1]特征字段是指属于同一类别的一组特征值，例如，特征字段“性别”包含两个值：“女性”和“男性”。

第 n^{th} 批次），奖励 R_{mn} 定义为过去 C 轮次中对应实例的平均损失值与当前轮次损失之间的差：

$$R_{mn} = \frac{1}{C}\sum_{c=1}^{C} L^{c}_{mn} - L_{mn} \tag{3}$$

其中，$R_{mn} > 0$ 表示该数据实例在当前轮次的损失小于其过去 C 轮次的平均损失值，这表明策略网络在当前轮次对该实例采取了更好的动作，反之亦然。为进行公平比较并提供改进激励，我们在每个训练轮次开始时初始化推荐系统模型的参数，并用最新轮次的损失值覆盖最早存储轮次的损失值。在此设置下，奖励可视为预测改进量。由于策略网络的参数不同，奖励能够反映优化结果，并用于调整这些参数。考虑到策略网络的动作采样行为会引入随机性，无法体现其实际选择能力，因此我们选择使用方程（3）中 L^{c}_{mn} 的平均值，使当前奖励 R_{mn} 更加可靠。

2.3 验证阶段

如训练阶段所述，推荐系统模型会在每个训练轮次中初始化，且通常尚未收敛。因此，我们设计了验证阶段，以全面评估策略网络的性能并收集最佳无噪声子集。为此，我们将验证阶段分为无噪声子集选择和推荐系统模型训练。对于子集选择，我们遍历整个训练集，并在每个小批次中选择置信度最高的数据实例 k，即采用 top-k 选择策略。获得选定子集后，我们初始化推荐系统模型的参数，并训练模型直至收敛，这与典型的推荐系统训练设置一致。最后，使用与搜索阶段相同的测试集和验证集，我们可以评估选定子集的质量以及训练后的策略网络的去噪能力[2]。只有测试性能最佳的策略网络和无噪声子集才会被采用。

个体选择与 ***Top-k*** 选择。与搜索阶段使用的个体选择相比，我们在验证阶段采用 top-k 选择，即针对每个小批次，由训练后的策略网络选出置信度最高的 k 个预测实例。采用这一方案的原因是，个体选择为策略网络优化提供了“局部”视角，而 top-k 选择则能提供“全局”视角，从而选出更具代表性的无噪声子集。从策略训练的角度来看，实例层面的“局部”视角有助于策略网络学习每个实例的有效性，并筛除噪声实例。然而，“局部”视角未必最适合决策，因为在模型训练期间，弱选择实例的贡献可能小于强选择实例，这可能会影响采用损失值平均值进行优化的模型。从这个角度来看，弱选择实例也可视为一种噪声，因此需要通过 top-k 选择策略施加“全局”约束。

[2]在实践中，我们在搜索阶段和验证阶段总共设置了50 个训练轮次，用于训练和评估策略网络。

算法 1 *Opt_I*：搜索阶段优化算法

输入：N数据小批次 $\boldsymbol{X} = \{\boldsymbol{X}^n_1, ..., \boldsymbol{X}^n_M\}^N_{n=1}$，策略网络 $\boldsymbol{\Theta}$，初始 RS 模型 $\boldsymbol{W}$，损失矩阵 $\boldsymbol{LMat}$
输出：训练后的策略网络 $\boldsymbol{\Theta}^*$ 和完整损失向量 $\boldsymbol{LVec}^*$

```
创建空的 LVec
初始化 W
对于所有 n ∈ [1, N]，执行
    s_n ← [X^n_1 ··· X^n_m ··· X^n_M]
    样本 a_1 ~ Θ(a | s_n)
    选择实例 X_slct1
    根据公式（4）估计 L_model(W)
    根据公式（3）和 LMat 进行估计 R_n
    根据公式（8）更新 Θ
    样本 a_2 ~ Θ(a | s)
    选择实例 X_slct2
    根据公式（4）估计 L_model(W)
    将 L_model(W) 追加到 LVec
    根据公式（5）更新 W
结束循环
Θ* ← Θ
```

2.4 优化方法

基于上述工作流程，本节将进一步详细介绍用于更新推荐系统模型和策略网络的优化方法。

2.4.1 推荐系统模型。由于搜索阶段和验证阶段的推荐系统模型相同，我们将模型参数和优化方法一并介绍。我们将推荐系统模型的参数记为 $\boldsymbol{W}$。在监督方式下，点击率预测可视为一个二元分类任务，其中输入实例包含特征和标签。因此，我们采用二元交叉熵损失函数 L_{model} 来优化 $\boldsymbol{W}$：

$$\mathcal{L}_{model}(\boldsymbol{W}) = -\frac{1}{M}\sum_{m=1}^{M}\left[y_m \log \hat{y}_m + (1-y_m)\log(1-\hat{y}_m)\right] \tag{4}$$

其中，y_m 和 $\hat{}y_m$ 分别是小批次中 m^{th} 实例的真实标签和推荐系统模型预测的概率。如第 2.2 节所述，损失值可视为搜索阶段所选输入实例的性能。通过最小化公式（4），我们可以得到训练良好的 $\boldsymbol{W}$，如下所示，其中 α 为学习率：

$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \alpha \nabla_{\boldsymbol{W}} \mathcal{L}_{model}(\boldsymbol{W}) \tag{5}$$

2.4.2 策略网络。由于这是一个强化学习框架，最优策略网络可以用 $\boldsymbol{\Theta}$ 参数化。给定奖励函数，即公式（3），需要优化的目标函数定义如下：

$$\max_{\boldsymbol{\Theta}} J(\boldsymbol{\Theta}) = \mathbb{E}_{a\sim\boldsymbol{\Theta}(a|s)}\left[R(a \mid s)\right] \tag{6}$$

其中 $a \sim \boldsymbol{\Theta}(a \mid s)$表示在状态 s下，通过策略$\boldsymbol{\Theta}(a \mid s)$采样得到动作 a。为了最大化上述函数，我们采用名为 REINFORCE 的策略梯度算法 [33]，，并采用蒙特卡洛采样来简化梯度的估计

算法 2 *Opt_II*：验证阶段优化算法

输入：N数据小批次 $\boldsymbol{X} = \{\boldsymbol{X}_1^n, ..., \boldsymbol{X}_M^n\}_{n=1}^N$，训练后的策略网络 $\boldsymbol{\Theta}^*$，初始 RS 模型 $\boldsymbol{W}$
输出：选定的数据子集 $\boldsymbol{X}_{sub}$，训练良好的推荐系统模型 $\boldsymbol{W}^*$

1：创建空对象 $\boldsymbol{X}_{sub}$
2：初始化 $\boldsymbol{W}$
3: **对所有** $n \in [1, N]$ **执行**
4: $s_n \leftarrow [\boldsymbol{X}_1^n \cdots \boldsymbol{X}_m^n \cdots \boldsymbol{X}_M^n]$
5: 样本 $a_3 \sim \boldsymbol{\Theta}^*(a \mid s_n)$
6: 选择实例 $\boldsymbol{X}_{slct3}$
7: 追加 $\boldsymbol{X}_{slct3}$ 至 $\boldsymbol{X}_{sub}$
8: **结束循环**
9: **当** 推荐系统模型 $\boldsymbol{W}$ 未收敛 **时**
10: 从 $\boldsymbol{X}_{sub}$中抽取一个数据批次
11: 根据公式（5）更新 $\boldsymbol{W}$
12: **结束 while 循环**

$\nabla_{\boldsymbol{\Theta}} J(\boldsymbol{\Theta})$，可表示为：

$$\begin{aligned}\nabla_{\boldsymbol{\Theta}} J(\boldsymbol{\Theta}) &= \sum_a R(a \mid s) \nabla \boldsymbol{\Theta}(a \mid s) \\ &= \sum_a R(a \mid s) \boldsymbol{\Theta}(a \mid s) \nabla \log \boldsymbol{\Theta}(a \mid s) \\ &= \mathbb{E}_{a \sim \boldsymbol{\Theta}(a|s)} [R(a \mid s) \nabla \log \boldsymbol{\Theta}(a \mid s)] \\ &\approx \frac{1}{T} \sum_{t=1}^{T} R(a \mid s) \nabla \log \boldsymbol{\Theta}(a \mid s)\end{aligned} \tag{7}$$

其中，T为样本数，我们在此处设置 $T = 1$ 以提高计算效率。参数 $\boldsymbol{\Theta}$可更新为：

$$\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} + \beta \nabla_{\boldsymbol{\Theta}} J(\boldsymbol{\Theta}) \tag{8}$$

其中，β为学习率。

2.4.3 搜索阶段优化算法。基于上述针对推荐系统模型和策略网络的优化方法，我们可以进行两阶段训练，以优化并评估所提出的框架。搜索阶段如算法 1 所示。在此阶段，我们遍历整个训练集，并交替优化 $\boldsymbol{\Theta}$ 和 $\boldsymbol{W}$，以获得表现良好的策略。为了公平比较奖励，我们在此阶段开始时初始化推荐系统模型 $\boldsymbol{W}$（第 2 行）。在训练轮次中，我们首先抽取一个数据实例小批次，作为策略网络的状态 s_n（第 4 行）。接下来，我们为批次中的每个数据实例抽取动作a_1（第 5 行），并将它们汇总为一个选择性批次 $\boldsymbol{X}_{slct1}$（第 6 行），随后将其输入推荐系统模型并获得相应的损失值（第 7 行）。利用该损失值以及先前轮次中保存的损失值[3]（即 $\boldsymbol{LMat}$），我们可以计算奖励（第 8 行）并更新策略网络（第 9 行）。类似地，推荐系统模型的优化（第 10–14 行）遵循与第 5 至第 9 行类似的流程。不过，我们会根据更新后的 $\boldsymbol{\Theta}$（第 10 行）抽取新的动作 a_2，并更新 $\boldsymbol{W}$（第 14 行），而不是像第 9 行那样更新 $\boldsymbol{\Theta}$。最后一个小批次中更新后的策略网络将作为训练良好的策略网络返回（第 16 行）。此外，为了记录损失值以供后续计算奖励基线，我们创建一个空的损失向量 $\boldsymbol{LVec}$（第 1 行），并在迭代过程中记录测试损失值（第 13 行）。

[3]在搜索阶段的前 C个轮次中，先前的C损失现已可用，因此我们在第 2.4.5 节中设计了一种预热训练方法。

算法 3 AutoDenoise 的整体优化算法

输入：N个数据小批次 $\boldsymbol{X} = \{\boldsymbol{X}_1^n, ..., \boldsymbol{X}_M^n\}_{n=1}^N$，预热轮次 C，训练轮次 T，空损失矩阵 $\boldsymbol{LMat} = [\boldsymbol{LVec}_1, ..., \boldsymbol{LVec}_C]$，初始 RS 模型 $\boldsymbol{W}$和策略网络 $\boldsymbol{\Theta}$
输出：训练良好的策略网络参数 $\boldsymbol{\Theta}^*$ 和选定的数据子集 $\boldsymbol{X}_{sub}$

1: **对于** $i = 1$； $i <= C$； $i++$ **执行**
2: 初始化 $\boldsymbol{W}$
3: **当** 推荐系统模型 $\boldsymbol{W}$ 尚未收敛 **时执行**
4: 抽取一个数据批次 $[\boldsymbol{X}_1^n \cdots \boldsymbol{X}_m^n \cdots \boldsymbol{X}_M^n]$
5: 根据公式（4）估计 $\mathrm{L}_{model}(\boldsymbol{W})$
6: 将 $\mathrm{L}_{model}(\boldsymbol{W})$ 添加到 $\boldsymbol{LVec}_i$
7: 根据公式（5）更新 $\boldsymbol{W}$
8: **结束 while 循环**
9: **结束 for 循环**
10: **对于** $t = 1$； $t <= T$； $t++$**执行**
11: $\boldsymbol{\Theta}^*, \boldsymbol{LVec}^* \leftarrow Opt_I(\boldsymbol{X}, \boldsymbol{\Theta}, \boldsymbol{W}, \boldsymbol{LMat})$
12: $\boldsymbol{LVec}_{t\%C} \leftarrow \boldsymbol{LVec}^*$
13: $\boldsymbol{X}_{sub}, \boldsymbol{W}^* \leftarrow Opt_II(\boldsymbol{X}, \boldsymbol{\Theta}^*, \boldsymbol{W})$
14: 测试 $\boldsymbol{W}^*$
15: **如果** $\boldsymbol{W}^*$ 表现最佳 **则**
16: 保存 $\boldsymbol{X}_{sub}$ 和 $\boldsymbol{\Theta}^*$
17: **结束如果**
18: **结束循环**

具体而言，来自上一个优化步骤的策略网络$\boldsymbol{\Theta}$无法表示当前数据批次经过优化后的性能。在此阶段，我们对每个小批次的数据使用两次，依次优化 $\boldsymbol{\Theta}$ 和 $\boldsymbol{W}$（第 9 行和第 14 行），从而避免干扰回传损失和优化环境。

2.4.4 验证阶段优化算法。算法2展示了验证阶段的完整流程，包括选择过程（第 3–8 行）以及对推荐系统模型进行完整训练的过程（第 9–12 行）。在此阶段，我们旨在通过训练推荐系统模型，使其在选定子集上收敛，从而评估训练后的策略网络 $\boldsymbol{\Theta}^*$ 的真实性能。具体而言，我们首先创建一个空集 $\boldsymbol{X}_{sub}$（第 1 行），并初始化 $\boldsymbol{W}$（第 2 行）。然后，我们使用 $\boldsymbol{\Theta}^*$遍历整个训练集以进行子集选择。与搜索阶段的流程类似，每个小批次都将经历采样（第 5 行）和选择（第 6 行）过程。随后，将得到的无噪声数据批次 $\boldsymbol{X}_{slct3}$ 添加到 $\boldsymbol{X}_{sub}$（第 7 行）。之后，我们将使用 $\boldsymbol{X}_{sub}$ 训练初始化后的 $\boldsymbol{W}$，直至其收敛（第 9–12 行）。我们将充分训练后的$\boldsymbol{W}^*$ 的性能视为验证结果。

2.4.5 整体优化算法。在介绍搜索阶段和验证阶段的优化算法后，本小节将进一步详细说明整体优化算法。完整算法如算法 3 所示。

预热训练。为了构建用于奖励计算的损失矩阵 $\boldsymbol{LMat}$，我们会在搜索阶段之前进行预热训练。具体而言，只有推荐系统模型会在以下条件下进行训练：

表1：不同骨干推荐模型的总体性能。

骨干模型	指标	FM		Wide & Deep		DCN		IPNN		DeepFM	
		w/o	**w**	**w/o**	**w**	**w/o**	**w**	**w/o**	**w**	**w/o**	**w**
MovieLens1M	曲线下面积 ↑	0.8080	**0.8095***	0.7957	**0.8022***	0.8092	**0.8115***	0.8023	**0.8063***	0.8048	**0.8097***
	对数损失 ↓	0.5334	**0.5292***	0.5400	**0.5333***	0.5235	**0.5219***	0.5433	**0.5363***	0.5306	**0.5225***
KuaiRec-Small	曲线下面积 ↑	0.8654	**0.8657**	0.8641	**0.8642**	0.8641	**0.8649**	0.8630	**0.8635**	0.8653	**0.8659**
	对数损失 ↓	0.4183	**0.4178**	0.4204	**0.4201**	0.4202	**0.4192***	0.4228	**0.4217***	0.4184	**0.4172***
Netflix	曲线下面积 ↑	0.6523	**0.6541***	0.6575	**0.6620***	0.6711	**0.6761***	0.6682	**0.6734***	0.6604	**0.6683***
	对数损失 ↓	0.7835	**0.7803***	0.9350	**0.9115***	0.7742	**0.7434***	0.7760	**0.7542***	0.8926	**0.8800***

"*"表示相较于**w/o** 版本取得了统计显著改进（即，带有 $p < 0.05$的双侧 t 检验）。

表 2：与基线方法的总体性能比较。（骨干模型：DeepFM）

数据集	指标	**w/o**	Drop3	BDIS	LSBo	LSSm	T-CE	R-CE	AutoDenoise
MovieLens1M	曲线下面积 ↑	0.8048	0.7701	0.6995	0.7659	0.7752	0.7814	0.7817	**0.8097***
	对数损失 ↓	0.5306	0.5776	0.7996	0.5851	0.5841	1.7886	1.8263	**0.5225***
KuaiRec-Small	曲线下面积 ↑	0.8653	0.8485	0.8018	0.8439	0.8566	0.8140	0.8189	**0.8659***
	对数损失 ↓	0.4184	0.4472	0.6694	0.4501	0.4931	1.7509	1.8955	**0.4172***
Netflix	曲线下面积 ↑	0.6604	0.6444	0.5726	0.6355	0.6471	0.6785	0.6832	0.6683
	对数损失 ↓	0.8926	1.3277	1.1192	1.4491	1.0330	2.5780	2.5801	**0.8800***

"*" 表示 相较于最佳基线方法的统计显著改进。↑：越高越好；↓：越低越好 .

与搜索阶段相同的设置，即在每轮次开始时初始化，并使用相同的损失函数，记为 L_{model}。迭代 C轮次后，我们便可得到一个 $\{C \times MN\}$ 矩阵，并以此开始搜索阶段。

整个优化流程包括预热训练（第 1–9 行）以及测试过程中的两个训练阶段（第 10–18 行）。在预热训练中，我们首先初始化推荐系统模型 $\boldsymbol{W}$ 的参数，并训练 C 轮使其收敛（第 1–9 行）；在此过程中，我们抽取一个小批次（第 4 行）进行模型优化（第 5 行），并保存相应的损失值（第 6 行），用于奖励计算。之后，我们按照算法 1（第 11 行）和算法2（第 13 行）训练并评估策略网络。利用选定子集优化后的 $\boldsymbol{W}^*$，我们最终可以评估其性能（第 14 行），并保存最优策略和子集（第 15–17 行）。经过总计 T 轮后保存的 $\boldsymbol{X}_{sub}$ 被视为无噪声子集。在实际推理阶段，我们用原始训练集替换该子集来进行模型训练。

3 实验

在本节中，我们在三个公开数据集上进行了大量实验，以探究 AutoDenoise 的有效性。

3.1 实验设置

3.1.1 数据集。我们在三个密度不同的公开数据集上评估 AutoDenoise 和基线方法的模型性能：**MovieLens1M**[4] (4.47%)、**KuaiRec-Small**[5] (99.62%) 和 **Netflix**[6] (0.02%)。对于所有数据集，我们随机选取 80% 作为训练集、10% 作为验证集，其余 10% 作为测试集。

3.1.2 评估指标。为公平地评估推荐性能，我们选取 AUC（受试者工作特征曲线下面积）分数和对数损失（损失值的对数）作为指标，这些指标广泛用于点击率预测任务 [3, 9]。在实践中，AUC 分数提高或对数损失值降低 0.001 量级，即表明性能显著提升 [3]。

3.1.3 实现细节。 AutoDenoise 基于一个 PyTorch 公共库7实现，该库包含十六种最先进的推荐系统模型。我们将 AutoDenoise 的策略网络开发为一个独立类，以便轻松地与不同的骨干推荐系统模型结合使用，如表 2 所示。具体而言，我们将其实现为一个两层 MLP，并额外添加嵌入层和输出层。MLP 中的每个全连接层都由一个线性层、一个批归一化操作、一个 *ReLU* 激活函数和一个随机失活操作（比率 $= 0.2$）组成。输出层采用 *softmax* 激活函数。对于超参数，我们将嵌入层的嵌入维度和两个全连接层的维度均设置为 $d = 16$。输出层的维度设为 2，以便为每个实例生成两个动作概率。

3.2 总体性能

本节评估 AutoDenoise 的有效性。如表1所示，我们比较了在经 AutoDenoise 去噪后的无噪声数据集（**使用**）上训练的推荐模型与在含噪声的原始数据集（**不使用**）上训练的模型的性能，并在相同的测试数据集上对它们进行评估。我们选择了多种先进的骨干推荐模型，包括因子分解机 [24], Wide & Deep [3], DCN [28], 和 DeepFM [9]。此外，我们还将 AutoDenoise 的去噪质量与

[4]https://grouplens.org/datasets/movielens/1m/ [5]
https://github.com/chongminggao/KuaiRec [6]
https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data

[7]https://github.com/rixwew/pytorch-fm

表3：可迁移性测试。

推荐模型	方法	MovieLens1M 曲线下面积 ↑	MovieLens1M 对数损失 ↓	KuaiRec-Small 曲线下面积 ↑	KuaiRec-Small 对数损失 ↓	Netflix 曲线下面积 ↑	Netflix 对数损失 ↓
FM	Normal	0.8080	0.5334	0.8654	0.4183	0.6523	0.7835
	AutoDenoise	**0.8082**	**0.5274***	**0.8657**	**0.4179**	**0.6532***	**0.7783***
Wide & Deep	Normal	0.7957	0.5400	0.8641	0.4204	0.6575	0.9350
	AutoDenoise	**0.8001***	**0.5363***	**0.8645**	**0.4202**	**0.6587***	**0.9212***
DCN	Normal	0.8092	0.5235	0.8641	0.4202	0.6711	0.7742
	AutoDenoise	**0.8111***	**0.5230**	**0.8648**	**0.4193**	**0.6762***	**0.7476***
IPNN	Normal	0.8023	0.5433	0.8630	0.4228	0.6682	0.7760
	AutoDenoise	**0.8029**	**0.5368***	**0.8644***	**0.4206***	**0.6723***	**0.7588***

“*” 表示 比最佳基线方法取得了统计显著改进。 ↑: 越高越好； ↓: 越低越好 .

最先进的实例选择方法（Drop3 [34], BDIS [2],LSBo [15], LSSm [15]）和实例去噪方法（T_CE [29], R_CE [29]）[8] 在表 2 中进行了比较。我们可以得出结论：

• 从表1的结果可以看出，整合 AutoDenoise 能够提升所有数据集上各骨干模型的性能。KuaiRec-Small 的数据密度远高于 MovieLens1M 和 Netflix（数据密度：99.62% ≫ 4.47% ＞ 0.02%），因此其改进不那么显著，可能是因为干净实例贡献占主导，而噪声实例对用户偏好建模的贡献较小；这是因为密集的用户-物品交互能够准确呈现用户的整体情况。相反，对于稀疏数据集 MovieLens1M 和 Netflix，AutoDenoise 得到的无噪声子集能够带来更大的性能提升。

• 在表 2 中，所有去噪方法都未能提升 DeepFM 的性能，这可能有两个原因。(i) 对于实例选择方法（DROP3 [34], BDIS [2], LSBo [15],LSSm [15]），它们往往会压缩数据集，导致丢弃过多样本并遗漏关键信息。(ii) 对于 T-CE 和 R-CE 这类去噪模型 [29],，它们虽然能提升高度稀疏数据集（Netflix，0.02%）上的 AUC 分数，却无法在密集数据集上带来收益。原因在于 T-CE 和 R-CE 高度依赖负采样策略，该策略会对未观测到的用户-物品交互进行采样。然而，对于 KuaiRec-Small 这类高度密集的数据集（99.62%），很难采样到有效的未观测交互。相反，

相反，凭借深度强化学习结构出色的探索能力，AutoDenoise 兼容各种密度的数据集。

3.3 可迁移性测试

在实际应用中，推荐器模型会频繁更新，因此为每个模型训练一个去噪框架既耗时又昂贵。因此，本节将考察 AutoDenoise 的可迁移性。

具体而言，在 DeepFM 的基础上，获得 AutoDenoise 选出的无噪声数据集后，我们直接用它训练其他骨干推荐模型（例如 FM [24],Wide & Deep [3], 和 DCN [28]），并考察它们的性能。结果见表3，其中“Normal”表示在含噪声的原始数据上训练，即表1中的“**w/o**”。我们可以观察到，所有骨干模型的性能都因在

[8]这两种基线方法可用于通用去噪任务。由于任务设置各不相同（如第3.1节所述），我们不测试其他去噪模型的性能。

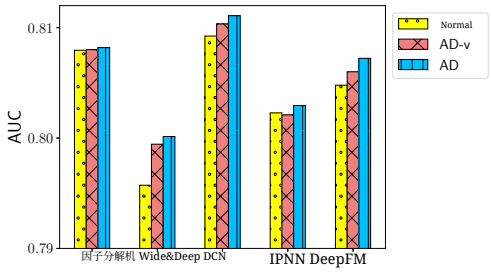

图3：MovieLens1M 上的消融实验。

AutoDenoise 选出的无噪声数据集上训练而得到提升。这些实验验证了 AutoDenoise 的输出具有很强的可迁移性，并有望应用于商业推荐系统。

3.4 消融实验

本节通过一系列实验考察 AutoDenoise 的核心组件。正如第2.3节所述，AutoDenoise 的一个关键部分是在验证阶段生成去噪结果的方式，即采用个体选择或 top-k 选择。因此，我们设计了一个名为**AutoDenoise-v**的变体，以探究不同选择策略的影响。与原始 AutoDenoise 相比，唯一的区别在于：AutoDenoise-v 在策略验证时采用个体选择，而 AutoDenoise 采用 Top-k 选择。

我们在 MovieLens1M 上测试了它们的 AUC 分数，结果如图3所示。可以看出，AutoDenoise (AD) 在所有模型上均优于 AutoDenoise-v (AD-v)，而 AutoDenoise-v 无法生成能够提升 IPNN 性能的无噪声数据集。这是因为 AutoDenoise-v 关注个体分数，却忽略了全局情况。此外，AutoDenoise 采用 top-k 选择策略，通过比较同一数据批次中的分数来捕获批次级全局信息，从而生成稳健的策略。该实验验证了我们模型设计的合理性。

3.5 参数分析

在本节中，我们使用 DeepFM 在 MovieLens1M 上进行实验，以研究两个关键超参数的敏感性：(i) 预热训练轮数 C；(ii) 选定样本数量 K。实验结果如图4所示，其中使用两个 Y 轴分别表示 AUC分数和对数损失值。图4 (a) 中的 X 轴表示

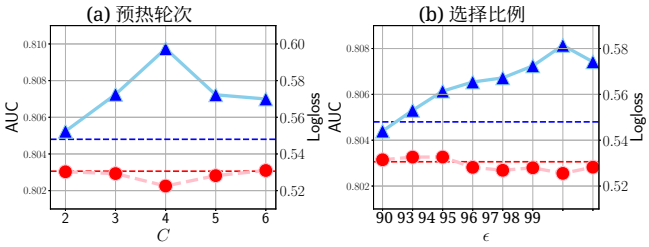

图4：MovieLens1M 上的参数分析。

预热轮次 C，图4 (b) 中的 X 轴表示选定比例 $\epsilon = K/M$。在两幅图中，蓝线表示 AUC 分数，红线表示对数损失值。此外，我们还用虚线标出"普通" DeepFM 的性能，以便进行直观比较（AUC 0.8048，对数损失 0.5306）。我们可以发现：

- 随着超参数变化，AutoDenoise 在大多数情况下均优于 Normal（$2 \le C \le 6$，$95\% \le \epsilon \le 99\%$），充分体现了其鲁棒性。在默认设置下（即 $C^* = 4$，$\epsilon^* = 98\%$），我们取得了最佳性能。

- 从图4（a）可以看出，参数 C 直接影响公式（3）中的奖励计算。对于 $C < C^*$，预热轮次过少会使 AutoDenoise 受到随机性的影响，即公式（3）中的第一项，从而导致性能下降。然而，预热轮次过多会使平均值趋于平滑，而每个批次的损失值（即公式（3）中的最后一项）对训练过程的影响则更为显著，导致奖励不稳定，并损害去噪准确率。

- 如图4（b）所示，恰当地丢弃实例至关重要。一方面，丢弃过多实例会因缺失有价值信息而导致性能较差。另一方面，如果保留过多实例，模型不可避免地会引入噪声，无法取得最佳性能。

3.6 案例研究

本节通过示例说明 AutoDenoise 的有效性。具体而言，我们从 MovieLens1M 中选取一位用户（ID：5051），并在表4中展示部分训练集，在表5中展示所有测试数据实例。对于训练集，我们展示电影的片名和类型，以及真实点击信号。对于测试数据，我们还展示了 DeepFM 在原始含噪数据（Normal）和无噪声数据（AutoDenoise）上训练后的预测结果 [9]。

在表4中，AutoDenoise 检测出《绿野仙踪》带有正向点击信号，属于噪声。这部电影是儿童片和歌舞片，而用户偏好的其他电影则主要是动作片、犯罪片和战争片。通过去除这条噪声，AutoDenoise 有助于修正 DeepFM 对《仙境》的预测（在表5中从 0.46 提高到 0.57）。这是一部关于战争的纪录片，也是该用户通常喜欢的类型。本案例研究也展示了 AutoDenoise 的可解释性潜力。

4 相关工作

本节将简要介绍与我们框架相关的工作，即实例选择和去噪。

实例选择。实例选择在挑选最具预测性的 数据实例方面发挥着重要作用，可在不降低预测模型性能的情况下缩减训练集 [20]。

表 4：用户 5051 的训练数据集。

标题	类型	点击
《爱国者》	动作|剧情|战争	1
《夺宝奇兵》	动作|冒险	1
《绿野仙踪》	**儿童|音乐**	**1**
...	...	...

表5：用户 5051 的测试数据集。

标题	类型	点击	普通/自动去噪
E.T.外星人	剧情|奇幻	1	0.77/0.82
阿拉伯的劳伦斯	冒险|战争	1	0.94/0.93
《仙境》	**纪录片|战争**	**1**	**0.46/0.57**

实例选择可分为两类，即包装方法 [1, 4, 10, 34],和过滤方法 [21, 22, 26], ，其选择标准分别基于模型性能和选择函数。CNN [10] 是实例选择领域最早提出的方法之一。它保留靠近类别边界的实例，并剔除内部实例，以生成有效的数据子集。DROP [34], 这一系列具有代表性的包装方法，依靠实例关联对象（即 k近邻中的 k近邻）来进行选择。LSSm 和 LSBo [15] 是基于局部集合 [1], 的方法，即包含同一类别样本的最大超球体。LSSm 将有用性和有害性定义为衡量标准，据此决定是否删除某个实例；LSBo 则依据类别边界作出决定。与以往的实例选择方法不同，AutoDenoise 使用策略网络自动区分噪声实例，而非采用固定的选择规则，因此能够更好地发现高度动态的数据分布中的内在关系。

去噪。 选择方式在去噪领域也表现良好。去噪方法可分为基于选择的方法 [6, 7, 32, 35], [29, 30] 和基于重加权的方法。WBPR [7] 为热门物品的缺失交互赋予更高的选择概率，因为它认为这些交互是真正的负实例。IR [32] 找出负交互，并更改其标签，以修正用户的不可靠行为。R-CE 和 T-CE [29] 认为噪声实例的损失值较高，因此分别采用两种策略：动态降低高损失实例的权重，并截断权重低于阈值的实例。然而，上述模型要么存在选择偏差，要么迁移能力不足，而 AutoDenoise 通过其深度强化学习框架解决了这些问题。

5 结论

我们提出了一种基于深度强化学习的实例去噪方法 AutoDenoise，通过在每个数据小批次中自适应地选择无噪声实例，并使用所选无噪声子集训练推荐系统模型，从而提升推荐系统模型的推荐性能。我们设计了一种两阶段优化策略，以恰当地训练和评估所提出的框架。大量实验验证了 AutoDenoise 及其选出的无噪声数据子集的有效性和兼容性。进一步的实验则证明了无噪声子集的可迁移性，即使用一个推荐系统模型选出的无噪声子集也能很好地迁移到其他最先进的骨干推荐系统模型，并显著提升其性能。

致谢

本研究部分获得了以下项目的资助：APRC - 香港城市大学新研究计划（编号：9610565，香港城市大学新入职教师启动基金）、SIRG - 香港城市大学战略跨学科研究基金（编号：7020046、7020074）、HKIDS青年学者研究基金（编号：9360163）、华为创新研究计划以及蚂蚁集团（CCF-蚂蚁研究基金）。

参考文献

[1] Henry Brighton 和 Chris Mellish。2002。基于实例的学习算法中的实例选择进展。数据挖掘与知识发现 6, 2 (2002), 153–172。 [2] Qingqiang Chen、Fuyuan Cao、Ying Xing 和 Jiye Liang。2022。实例选择：贝叶斯决策理论视角。*AAAI*人工智能会议论文集 36, 6, 6287 6294 10 1609 36 6 20578 – 。 [3] https://doi.org/ 。 /aaai.v i。Heng-Tze Cheng、Levent Koc、Jeremiah Harmsen、Tal Shaked、Tushar Chandra、Hrishi Aradhye、Glen Anderson、Greg Corrado、Wei Chai、Mustafa Ispir 等。2016。面向推荐系统的 Wide & Deep 学习。载于首届深度学习与推荐系统研讨会论文集。7–10。 [4] Chien-Hsing Chou、Bo-Han Kuo 和 Fu Chang。2006。广义压缩最近邻规则作为一种数据约简方法。载于第*18*届国际模式识别会议（*ICPR'06*），第2卷。IEEE，556–559。 [5] James Davidson、Benjamin Liebald、Junning Liu、Palash Nandy、Taylor Van Vleet、Ullas Gargi、Sujoy Gupta、Yu He、Mike Lambert、Blake Livingston 等。2010。YouTube 视频推荐系统。载于第四届*ACM*推荐系统会议论文集。293–296。 [6] Jingtao Ding、Guanghui Yu、Xiangnan He、Fuli Feng、Yong Li 和 Depeng Jin。2019。利用浏览数据设计贝叶斯个性化排序采样器。*IEEE*知识与数据工程汇刊 33, 2 (2019), 667–681。 [7] Zeno Gantner、Lucas Drumond、Christoph Freudenthaler 和 Lars Schmidt-Thieme。2012。针对非均匀采样物品的个性化排序。载于*KDD Cup 2011*论文集。PMLR，231–247。 [8] Zhabiz Gharibshah、Xingquan Zhu、Arthur Hainline 和 Michael Conway。2020。在线展示广告中用户兴趣与响应预测的深度学习。数据科学与工程 5, 1 (2020), 12–26。 [9] Huifeng Guo、Ruiming Tang、Yunming Ye、Zhenguo Li 和 Xiuqiang He。2017。DeepFM：一种基于因子分解机的点击率预测神经网络。载于*IJCAI*论文集。[10] Peter Hart。1968。压缩最近邻规则（通信）。*IEEE*信息论汇刊 14, 3 (1968), 515–516。 [11] Kaixi Hu、Lin Li、Qing Xie、Jianquan Liu 和 Xiaohui Tao。2021。当序列预测遇上隐式困难交互，接下来会怎样？载于第*30*届*ACM*信息与知识管理国际会议论文集。710–719。 [12] Yifan Hu、Yehuda Koren 和 Chris Volinsky。2008。面向隐式反馈数据集的协同过滤。载于*2008*年第八届*IEEE*国际数据挖掘会议。IEEE，263–272。 [13] PN Vijaya Kumar 和 V Raghunatha Reddy。推荐系统（RSS）及其应用综述。国际计算机与通信工程创新研究期刊 2, 8 (2014), 5254–5260。 [14] Dongha Lee、SeongKu Kang、Hyunjun Ju、Chanyoung Park 和 Hwanjo Yu。2021。为单类协同过滤引导式学习用户和物品表示。载于第*44*届*ACM SIGIR*国际信息检索研究与发展会议论文集。317–326。 [15] Enrique Leyva、Antonio González 和 Raúl Pérez。2015。基于局部集合的三种新实例选择方法：从双目标视角与多种方法进行比较研究。模式识别 48, 4 (2015), 1523–1537。 [16] Tzu-Heng Lin、Chen Gao 和 Yong Li。2019。Cross：面向社交电子商务的跨平台推荐。载于第*42*届*ACM SIGIR*国际信息检索研究与发展会议论文集。515–524。 [17] Weilin Lin、Xiangyu Zhao、Yejing Wang、Tong Xu 和 Xian Wu。2022。AdaFS：深度推荐系统中的自适应特征选择。载于第*28*届*ACM SIGKDD*知识发现与数据挖掘大会论文集。3309–3317。 [18] Haochen Liu、Xiangyu Zhao、Chong Wang、Xiaobing Liu 和 Jiliang Tang。2020。深度推荐系统中的自动嵌入维度搜索。载于第*43*届*ACM SIGIR*国际信息检索研究与发展会议论文集。2307–2316。 [19] Kunpeng Liu、Yanjie Fu、Pengfei Wang、Le Wu、Rui Bo 和 Xiaolin Li。2019。通过多智能体强化学习自动探索特征子空间。载于第*25*届*ACM SIGKDD*国际知识发现与数据挖掘会议论文集。207–215。 [20] J Arturo Olvera-López、J Ariel Carrasco-Ochoa、J Martínez-Trinidad 和 Josef Kittler。2010。实例选择方法综述。人工智能评论 34, 2 (2010), 133–143。 [21] J Arturo Olvera-López、J Ariel Carrasco-Ochoa 和 J Fco Martínez-Trinidad。2008。通过原型相关性进行原型选择。载于伊比利亚美洲模式识别大会。Springer，153–160。 [22] Thanapant Raicharoen 和 Chidchanok Lursinsap。2005。针对成对异类最近邻（POC-NN）算法的分而治之方法。《模式识别快报》26, 10 (2005), 1554–1567。 [23] Hassan Ramchoun、Youssef Ghanou、Mohamed Ettaouil 和 Mohammed Amine Janati Idrissi。2016。多层感知机：架构优化与训练。（2016）。 [24] Steffen Rendle。2010。因子分解机。载于2010年IEEE国际数据挖掘会议。IEEE，995–1000。 [25] Francesco Ricci、Lior Rokach 和 Bracha Shapira。2011。推荐系统手册简介。载于《推荐系统手册》。Springer，1–35。 [26] José C Riquelme、Jesús S Aguilar-Ruiz 和 Miguel Toro。2003。利用有序投影寻找代表性模式。《模式识别》36, 4 (2003), 1009–1018。 [27] Jian Wang 和 Yi Zhang。2013。电子商务推荐机会模型：合适的产品，合适的时间。载于第36届ACM国际信息检索研究与发展大会论文集。303–312。 [28] Ruoxi Wang、Bin Fu、Gang Fu 和 Mingliang Wang。2017。用于广告点击预测的深度交叉网络。载于ADKDD'17论文集。1–7。 [29] Wenjie Wang、Fuli Feng、Xiangnan He、Liqiang Nie 和 Tat-Seng Chua。2021。对用于推荐的隐式反馈进行去噪。载于第14届ACM国际网络搜索与数据挖掘会议论文集。373–381。 [30] Yu Wang、Xin Xin、Zaiqiao Meng、Joemon M Jose、Fuli Feng 和 Xiangnan He。2022。通过跨模型一致性学习鲁棒推荐系统。载于2022年ACM网络会议论文集。2015–2025。 [31] Yejing Wang、Xiangyu Zhao、Tong Xu 和 Xian Wu。2022。Autofield：深度推荐系统中的特征选择自动化。载于2022年ACM网络会议论文集。1977–1986。 [32] Zitai Wang、Qianqian Xu、Zhiyong Yang、Xiaochun Cao 和 Qingming Huang。2021。隐式反馈并非总是有利：针对噪声交互的迭代重标记单类协同过滤。载于第29届ACM国际多媒体会议论文集。3070–3078。 [33] Ronald J Williams。1992。用于连接主义强化学习的简单统计梯度跟随算法。《机器学习》8, 3 (1992), 229–256。 [34] D Randall Wilson 和 Tony R Martinez。2000。基于实例的学习算法的数据约简技术。《机器学习》38, 3 (2000), 257–286。 [35] Wenhui Yu 和 Zheng Qin。2020。通过噪声标签鲁棒学习为隐式反馈数据设计采样器。载于第43届ACM SIGIR国际信息检索研究与发展会议论文集。861–870。 [36] Fajie Yuan、Xin Xin、Xiangnan He、Guibing Guo、Weinan Zhang、Chua Tat-Seng 和 Joemon M Jose。2018。fBGD：利用BGD从正例未标记数据中学习嵌入。（2018）。 [37] Chi Zhang、Yantong Du、Xiangyu Zhao、Qilong Han、Rui Chen 和 Li Li。2022。面向序列去噪的分层物品不一致信号学习。载于第31届ACM信息与知识管理国际会议论文集。2508–2518。 [38] Xiangyu Zhao。2022。自适应与自动化深度推荐系统。《ACM SIGWEB通讯》2022年春季刊（2022），1–4。 [39] Xiangyu Zhao、Changsheng Gu、Haoshenglun Zhang、Xiwang Yang、Xiaobing Liu、Hui Liu 和 Jiliang Tang。2021。DEAR：推荐系统中的在线广告曝光深度强化学习。载于AAAI人工智能会议论文集，第35卷。750–758。 [40] Xiangyu Zhao、Haochen Liu、Wenqi Fan、Hui Liu、Jiliang Tang 和 Chong Wang。2021。AutoLoss：推荐中的自动化损失函数搜索。载于第27届ACM SIGKDD知识发现与数据挖掘大会论文集。3959–3967。 [41] Xiangyu Zhao、Long Xia、Jiliang Tang 和 Dawei Yin。2019。面向搜索、推荐和在线广告的深度强化学习：综述。《ACM SIGWEB通讯》2019年春季刊（2019），1–15。 [42] Xiangyu Zhao、Long Xia、Liang Zhang、Zhuoye Ding、Dawei Yin 和 Jiliang Tang。2018。面向逐页推荐的深度强化学习。载于第12届ACM推荐系统会议论文集。ACM，95–103。 [43] Xiangyu Zhao、Liang Zhang、Zhuoye Ding、Long Xia、Jiliang Tang 和 Dawei Yin。2018。通过成对深度强化学习利用负反馈进行推荐。载于第24届ACM SIGKDD知识发现与数据挖掘大会论文集。ACM，1040–1048。 [44] Xiangyu Zhao、Liang Zhang、Zhuoye Ding、Dawei Yin、Yihong Zhao 和 Jiliang Tang。2017。面向列表式推荐的深度强化学习。arXiv预印本 arXiv:1801.00209（2017）。 [45] Guorui Zhou、Xiaoqiang Zhu、Chenru Song、Ying Fan、Han Zhu、Xiao Ma、Yanghui Yan、Junqi Jin、Han Li 和 Kun Gai。2018。用于点击率预测的深度兴趣网络。载于第24届ACM SIGKDD国际知识发现与数据挖掘会议论文集。1059–1068。