文章编号：2095-2163(2022)05-0001-10 中图分类号：TP391 文献标志码：A

# 融合实例频率加权正则化和神经网络的概率矩阵分解模型

韦凯欣[1]，宋 燕[2]
（1 上海理工大学 理学院，上海 200093；2 上海理工大学 光电信息与计算机工程学院，上海 200093）

**摘 要：** 针对大规模稀疏数据集上基于概率矩阵分解(Probabilistic Matrix Factorization，PMF)模型的缺失数据估计方法容易出现过拟合且迭代训练时间较长的问题，本文创新性地提出一种融合实例频率加权正则化(Instance-frequency-weighted Regularization，IR)和神经网络(Neural Network，NN)的概率矩阵分解(IR-NNPMF)模型。一方面，充分考虑已知数据在数据集中的不平衡分布现象，并根据潜在因子相关的实例频率对其正则化参数进行加权处理，从而提高模型的泛化能力和预测精度；另一方面，引入神经网络来缓解模型迭代训练造成的时间消耗，提高模型预测精度的同时减少模型训练时间。最后，在 4 个真实数据集上的实验结果证明，和带标准 $L_2$ 正则化和使用传统梯度下降训练算法的经典 PMF 模型相比，本文提出的 IR-NNPMF 模型能够在提高预测精度的同时大大提高模型的训练速度。

**关键词：** 概率矩阵分解；大规模稀疏数据；不均衡分布；正则化；神经网络

## Probabilistic matrix factorization model based on instance frequency weighted regularization and neural networks

WEI Kaixin[1], SONG Yan[2]
(1 College of Science, University of Shanghai for Science and Technology, Shanghai 200093, China; 2 School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China)

**【Abstract】** As the missing data estimation method based on Probabilistic Matrix Factorization (PMF) model on large-scale sparse data is prone to overfitting and long iterative training time, this paper innovatively proposes a so-called instance-frequency-weighted regularization incorporated and neural network PMF (IR-NNPMF) model that combines instance-frequency-weighted Regularization (IR) and Neural Network (NN). On the one hand, the unbalanced distribution of known data in the data matrix is fully considered to improve the generalization ability and prediction accuracy of the model, and the regularization parameters are weighted according to the instance frequency related to latent factors. On the other hand, neural network is introduced to alleviate the time consumption caused by iterative training, which can improve the prediction accuracy of the model while reducing the model training time. Finally, experimental results on four real data sets show that compared with the classical PMF models with standard $L_2$ regularization or traditional gradient descent training algorithm, the proposed IR-NNPMF model can greatly improve the prediction accuracy and training speed of the model.

**【Key words】** probabilistic matrix factorization; large-scale sparse data; imbalanced distribution; regularization scheme; neural network

## 0 引 言

近年来，随着计算机技术的全面发展，信息呈现爆炸式增长。面对这些海量数据，如何通过数据挖掘相关理论和技术手段挖掘出潜在的、有效的以及可被解释和理解的信息，一直以来都是亟待解决的问题[1-2]。矩阵分解(Matrix Factorization，MF)模型，特别是概率矩阵分解(Probabilistic Matrix Factorization，PMF)模型由于其在大规模稀疏数据集上的良好性能而被广泛应用。通过将原始的高维数据矩阵映射到低维的潜在因子空间，来挖掘隐藏的有用信息[3-4]。然而，由于天然的技术手段不足或者后期存储设备故障等原因，数据缺失难以避免，这就造成了原始数据矩阵的高维稀疏现象。当面对大规模、不均衡稀疏数据时，PMF 模型往往会出现过拟合和迭代训练时间复杂度高等问题。

在大规模、不均衡稀疏矩阵上进行矩阵分解，本质上是一个不适定问题，难以找到唯一的或者全局最优解。正则化通常是解决上述问题，提高模型泛化能力的有效方法之一[5-8]。然而，目前大部分矩阵分解的方法都采用了一般的正则化方法，如 $L_2$ 正则化[6,8-11]。但是，这些正则化方法通常是针对没有缺

基金项目：国家自然科学基金(62073223)；中央军委装备发展部航天飞行动力学技术国防科技重点实验室资助(6142210200304)。

作者简介：韦凯欣(1997-)，女，硕士研究生，主要研究方向：推荐系统、数据挖掘；宋 燕(1979-)，女，博士，教授，博士生导师，CCF 高级会员(No.93073SM)，主要研究方向：大数据算法、图像处理、预测控制。

通讯作者：宋 燕 Email：songya@usst.edu.com

收稿日期：2021-12-30

失项的完整数据集的,高维稀疏数据集在稀疏度和数据均衡度方面都存在很大的差异,直接使用这些传统的正则化方法取得的效果可能事倍功半。如何设计出适用于不完整数据集的正则化方案,更加准确地刻画原始矩阵,已然成为亟待解决的重要课题。

另外,时间复杂度也是判断一个模型性能好坏的标准之一。目前,大部分模型的求解方法,都依赖于迭代求解,如梯度下降法(Gradient Descent, GD)。其主要思想是沿着参数当前位置的负梯度方向进行迭代更新,逐步达到最优解[12-14]。然而,这种训练方法,往往需要付出巨大的时间代价,而且随着数据集的不断扩大,这种时间消耗更是难以忍受。不同于这些传统的训练方法,基于神经网络的方法通常仅需一次迭代,可以大大提升模型的训练效率。因此,本文将神经网络引入到模型的训练过程当中。

综上所述,本文提出一种融合实例频率加权正则化和神经网络的概率矩阵分解模型,在提高模型预测精度的同时大大提升模型训练效率。本文的主要贡献如下:

(1)基于传统的 PMF 模型,建立一种新的融合数据不均衡分布信息和神经网络的概率矩阵分解模型,即 IR-NNPMF 模型。

(2)基于稀疏矩阵行列之间数据不均衡分布的信息,通过引入实例频率加权的正则化方案,有效地缓解模型过拟合问题。

(3)引入神经网络进行模型的参数训练,整个训练过程当中,只需要进行一次迭代,大大提高了模型的训练效率。

(4)在 4 个真实的工业应用数据集上的实验结果证明,本文提出的 IR-NNPMF 模型可以有效缓解模型的过拟合问题,在提升预测精度的同时减少模型的训练时间。

## 1 相关工作

### 1.1 概率矩阵分解模型

概率矩阵分解主要是通过 2 个低维潜在因子矩阵相乘去近似原始的高维稀疏矩阵[9]。假设原始的高维稀疏矩阵为 $\boldsymbol{R} \in \mathbb{R}^{N\times M}$,得到的近似矩阵为 $\hat{\boldsymbol{R}} = \boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}$,其中 $\boldsymbol{U} \in \mathbb{R}^{N\times D}$, $\boldsymbol{V} \in \mathbb{R}^{M\times D}$,并且 $D \ll \min\{N,M\}$。

为了对模型进行学习,首先,假设已经观测到的原始矩阵 $\boldsymbol{R}$ 服从以下高斯分布:

$$P(\boldsymbol{R} \mid \boldsymbol{U},\boldsymbol{V},\sigma^2) = \prod_{i=1}^{N}\prod_{j=1}^{M} [N(R_{ij} \mid \boldsymbol{U}_i \boldsymbol{V}_j^{\mathrm{T}},\sigma_R^2)]^{I_{ij}^R} \quad (1)$$

其中,$N(x \mid \mu,\sigma^2)$ 是均值为 $\mu$,方差为 $\sigma^2$ 的高斯分布的概率密度函数;$R_{ij}$ 表示位于矩阵 $\boldsymbol{R}$ 中第 $i$ 行和第 $j$ 列的元素;$\boldsymbol{U}_i$ 表示位于矩阵 $\boldsymbol{U}$ 中第 $i$ 行的行向量;$\boldsymbol{V}_j$ 表示位于矩阵 $\boldsymbol{V}$ 中第 $j$ 行的行向量;$I_{ij}$ 是指示函数,如果 $R_{ij}$ 有数据,则为 1,否则为 0。

为了防止过拟合,假设 2 个低秩矩阵的潜在因子向量服从零均值的高斯先验分布:

$$P(\boldsymbol{U} \mid \sigma_U^2) = \prod_{i=1}^{N} N(\boldsymbol{U}_i \mid 0,\sigma_U^2\boldsymbol{I}) \quad (2)$$

$$P(\boldsymbol{V} \mid \sigma_V^2) = \prod_{j=1}^{M} N(\boldsymbol{V}_j \mid 0,\sigma_V^2\boldsymbol{I}) \quad (3)$$

根据贝叶斯公式,可得潜在因子向量的后验 log 函数:

$$log(\boldsymbol{U},\boldsymbol{V} \mid \boldsymbol{R},\sigma^2,\sigma_U^2,\sigma_V^2) = -\frac{1}{2\sigma^2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}^R(R_{ij} - \boldsymbol{U}_i \boldsymbol{V}_j^{\mathrm{T}})^2 - \frac{1}{2\sigma_U^2}\sum_{i=1}^{N} \boldsymbol{U}_i \boldsymbol{U}_i^{\mathrm{T}} - \frac{1}{2\sigma_V^2}\sum_{j=1}^{M} \boldsymbol{V}_j \boldsymbol{V}_j^{\mathrm{T}} - \frac{1}{2}\left(\left(\sum_{i=1}^{N}\sum_{j=1}^{M} (I_{ij}^R)^2\right)\ln\sigma^2 + ND\ln\sigma_U^2 + MD\ln\sigma_V^2\right) + C \quad (4)$$

其中,$C$ 是不依赖于任何参数的常量。

最大化 $log$ 函数等价于最小化式(5)中的目标函数,即:

$$J = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}^R (R_{ij} - \boldsymbol{U}_i \boldsymbol{V}_j^{\mathrm{T}})^2 + \frac{\lambda_U}{2}\|\boldsymbol{U}_i\|_{Fro}^2 + \frac{\lambda_V}{2}\|\boldsymbol{V}_j\|_{Fro}^2 \quad (5)$$

其中,$\|\cdot\|_{Fro}^2$ 表示 Frobenius 范数;$\lambda_U = \frac{\sigma_U^2}{\sigma^2}$;$\lambda_V = \frac{\sigma_V^2}{\sigma^2}$。

### 1.2 实例频率加权正则化方案

不均衡高维稀疏数据集上的不适定问题往往严重依赖于最初的假设[6,8-11]。加入正则化是解决这类问题的有效方法之一[6-8],例如 $L_2$ 正则化。式(5)的 $L_2$ 正则化方案为:

$$L = \frac{\lambda_U}{2}\|\boldsymbol{U}_i\|_{Fro}^2 + \frac{\lambda_V}{2}\|\boldsymbol{V}_j\|_{Fro}^2 \quad (6)$$

然而,这些传统的正则化方法通常适用于完整的数据矩阵。面对高维稀疏的数据矩阵,就需要根据每行或每列不同的已知实例数,为其分配不同的正则化系数。因此,将式(6)改进为式(7):

$$L(\boldsymbol{U},\boldsymbol{V}) = \frac{\gamma}{2}\left(\sum_{i=1}^{N} |\Lambda(\boldsymbol{U}_i)|^{\beta}\|\boldsymbol{U}_i\|_{Fro}^2 +

$$\sum_{j=1}^{M} |\Lambda(\boldsymbol{V}_j)|^{\beta} \|\boldsymbol{V}_j\|_{Fro}^2\Big) \tag{7}$$

其中，$\Lambda(\boldsymbol{U}_i)$ 表示行向量 $\boldsymbol{U}_i$ 中的已知实例的个数；$\Lambda(\boldsymbol{V}_j)$ 表示行向量 $\boldsymbol{V}_j$ 中已知实例的个数；$f(x) = x^{\beta}$ 表示底数为 $x$，指数为 $\beta$ 的幂函数，用来实现对正则化效果的细粒度控制。

### 1.3　自编码神经网络

自编码神经网络模型作为一种无监督模型，通过重构输入信号，提取出比原始无标注数据更好的特征表述，从而取得更好的模型效果[15]。最简单的自编码神经网络包括 3 层：输入层、隐藏层（也称编码层）和输出层（也称解码层）[16-17]。具体的自编码神经网络结构图如图 1 所示。

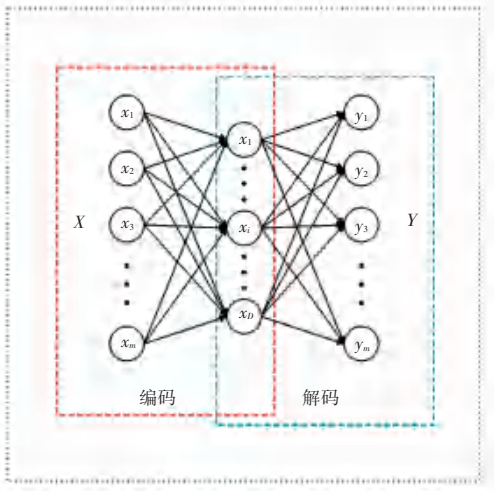


**图 1　自编码神经网络模型**

**Fig. 1　Autoencoder Neural Network model**

假设输入层有 $m$ 个样本 $x_1, x_2, \cdots, x_m$，隐藏层有 $D$ 个神经元，对应的输出层为 $y_1, y_2, \cdots, y_m$。首先，通过隐藏层将原始输入数据映射到更低维度的空间，从而提取出重要特征。此处需用到的数学公式为：

$$h = f(\boldsymbol{w}_1 \cdot \boldsymbol{x} + \boldsymbol{b}_1) \tag{8}$$

其中，$\boldsymbol{w}_1$ 是权重参数矩阵；$\boldsymbol{b}_1$ 是偏置参数向量；$f$ 是激活函数，例如 $f(x) = 1/(1 + \exp(-x))$。

其次，再通过输出层重构这些重要特征。相应的数学公式可表示为：

$$\boldsymbol{y} = f(\boldsymbol{w}_2 \cdot h + \boldsymbol{b}_2) \tag{9}$$

其中，$\boldsymbol{w}_2$ 是权重参数矩阵；$\boldsymbol{b}_2$ 是偏置参数向量；$f$ 同样是激活函数。为了能更好地表述原始数据，输出层通常与输入层的维度一致。

最后，构建从输入层到输出层的代价函数，并最小化代价函数得到最优特征表述。由此推得的数学公式为：

$$C = \frac{1}{2}\sum_{i=1}^{n} \|x_i - y_i\|_F^2 \tag{10}$$

其中，$n$ 表示输入层样本的个数。

## 2　主要结果

### 2.1　IR-NNPMF 模型

面对大规模稀疏数据集上的数据不均衡分布现象，传统的正则化方法取得的效果并不理想。因此，本文创造性地提出一种融合实例频率加权正则化和神经网络的 PMF 模型，即 IR-NNPMF 模型。具体的模型结构图如图 2 所示。

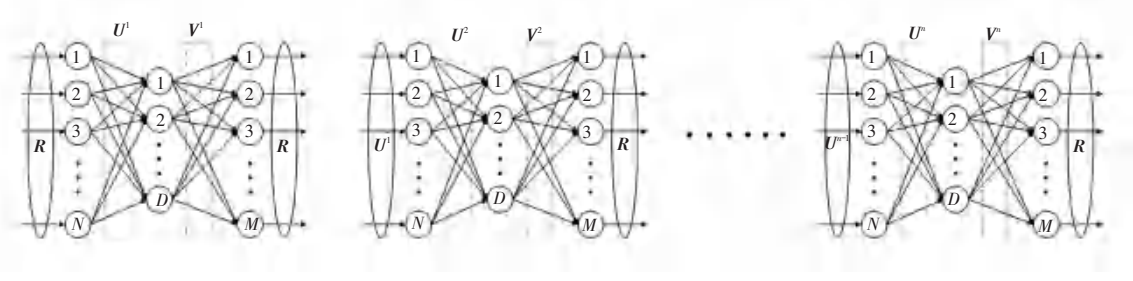


**图 2　IR-NNPMF 模型结构图**

**Fig. 2　Structure of an IR-NNPMF model**

本文将与潜在因子有关的实例频率融入到正则化项当中，故对传统 PMF 模型当中的潜在因子的先验分布条件进行改进，具体如下所示：

$$P(\boldsymbol{U} \mid \sigma_U^2) = \prod_{i=1}^{N} N(\boldsymbol{U}_i \mid 0, |\Lambda(\boldsymbol{U}_i)|^{\beta}\sigma_U^2\boldsymbol{I}) \tag{11}$$

$$P(\boldsymbol{V} \mid \sigma_V^2) = \prod_{j=1}^{M} N(\boldsymbol{V}_j \mid 0, |\Lambda(\boldsymbol{V}_j)|^{\beta}\sigma_V^2\boldsymbol{I}) \tag{12}$$

潜在因子的对数后验分布函数如式（13）所示：

$$log(\boldsymbol{U},\boldsymbol{V}\mid R,\sigma^2,\sigma_U^2,\sigma_V^2) = -\frac{1}{2\sigma^2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}^R(R_{ij} - \boldsymbol{U}_i\boldsymbol{V}_j^{\mathrm{T}})^2 - \frac{1}{2\sigma_U^2}\sum_{i=1}^{N}\frac{1}{|\Lambda(U_i)|^{\beta}}\boldsymbol{U}_i\boldsymbol{U}_i^{\mathrm{T}} - \frac{1}{2\sigma_V^2}\sum_{j=1}^{M}\frac{1}{|\Lambda(\boldsymbol{V}_j)|^{\beta}}\boldsymbol{V}_j\boldsymbol{V}_j^{\mathrm{T}} + \ln(2\pi\sigma^2)^{-1/2}\Big(\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}^R\Big) + \sum_{i=1}^{N}\ln(2\pi\sigma_U^2|\Lambda(\boldsymbol{U}_i)|^{\beta})^{-\frac{1}{2}} + \sum_{j=1}^{M}\ln(2\pi\sigma_V^2|\Lambda(\boldsymbol{V}_j)|^{\beta})^{-\frac{1}{2}} + C \tag{13}$$

最终的目标函数可表示为：

$$\min J = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}^R(R_{ij} - \boldsymbol{U}_i\boldsymbol{V}_j^{\mathrm{T}})^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N}|\Lambda(\boldsymbol{U}_i)|^{-\beta}\|\boldsymbol{U}_i\|_{Fro}^2 + \frac{\lambda_V}{2}|\Lambda(\boldsymbol{V}_j)|^{-\beta}\|\boldsymbol{V}_j\|_{Fro}^2 \tag{14}$$

## 2.2 基于神经网络进行参数训练

本文引入神经网络进行参数训练，提高模型的训练效率。假设 IR-NNPMF 模型包含 $n$ 层神经网络，研究展开的模型训练过程为：

(1)当 $n=1$ 时，即模型包含一层神经网络。将矩阵 $\boldsymbol{R}$ 的每一行输入到输入层，并分别通过潜在因子矩阵 $\boldsymbol{U}^1$ 和 $\boldsymbol{V}^1$ 对输入数据进行编码和解码。

(2)当 $n\geqslant 2$ 时，即模型包含多层神经网络。对于第一层神经网络，其训练过程和单层神经网络模型相同；在第 2 层及之后的训练过程中，将上一层输出的潜在因子矩阵 $\boldsymbol{U}^{n-1}$ 的每一行向量作为输入数据，进行当前层的 $\boldsymbol{U}^n$ 的训练。

紧接着，将对模型参数 $\boldsymbol{U}^n$ 和 $\boldsymbol{V}^n$ 进行求解。

(1)当 $n=1$ 时。研发求解过程如下。

首先，初始化潜在因子矩阵 $\boldsymbol{U}^1$。将原始矩阵 $\boldsymbol{R}$ 进行输入，然后通过随机学习算法[18-20]得到权重矩阵 $\boldsymbol{W}^{D\times M}$ 和偏置向量 $\boldsymbol{b}^{1\times D}$，之后首次更新 $\boldsymbol{U}^1$：

$$\boldsymbol{U}^1 = \begin{bmatrix} u_1^1 \\ u_2^1 \\ \vdots \\ u_N^1 \end{bmatrix} = \begin{bmatrix} h(\boldsymbol{w}_1\boldsymbol{R}_1^{\mathrm{T}} + b_1) & \cdots & h(\boldsymbol{w}_D\boldsymbol{R}_1^{\mathrm{T}} + b_D) \\ \vdots & \ddots & \vdots \\ h(\boldsymbol{w}_1\boldsymbol{R}_N^{\mathrm{T}} + b_1) & \cdots & h(\boldsymbol{w}_D\boldsymbol{R}_N^{\mathrm{T}} + b_D) \end{bmatrix} \tag{15}$$

其中，$\boldsymbol{w}_1,\boldsymbol{w}_2,\cdots,\boldsymbol{w}_D$ 表示权重矩阵 $\boldsymbol{W}^{D\times M}$ 的每一行元素；$b_1,b_2,\cdots,b_D$ 表示偏置向量 $\boldsymbol{b}^{1\times D}$ 的每一个元素。

其次，更新潜在因子矩阵 $\boldsymbol{V}^1$。具体做法如下：

① 计算 $\boldsymbol{V}^1$ 中每一个行向量的偏导数：

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{V}_j^1} = -\sum_{i=1}^{N} I_{ij}^R(\boldsymbol{U}_i^1)^{\mathrm{T}}(R_{ij} - \boldsymbol{U}_i^1(\boldsymbol{V}_j^1)^{\mathrm{T}}) + \lambda_V|\Lambda(\boldsymbol{V}_j)|^{-\beta}(\boldsymbol{V}_j^1)^{\mathrm{T}} \tag{16}$$

② 令 $\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{V}_j^1} = 0$，整理可得更新公式为：

$$(\boldsymbol{V}_j^1)^{\mathrm{T}} = \left(\sum_{i=1}^{N} I_{ij}^R(\boldsymbol{U}_i^1)^{\mathrm{T}}(\boldsymbol{U}_i^1) + \lambda_V|\Lambda(\boldsymbol{V}_j)|^{-\beta}\boldsymbol{E}\right)^{-1}\left[\sum_{i=1}^{N} I_{ij}^R(\boldsymbol{U}_i^1)^{\mathrm{T}}R_{ij}\right] \tag{17}$$

其中，$\boldsymbol{E}$ 表示一个维度为 $D\times D$ 的单位矩阵。

最后，再次更新潜在因子矩阵 $\boldsymbol{U}^1$。首次更新的 $\boldsymbol{U}^1$ 是通过随机学习算法和神经网络得到的，具有一定的随机性。为了更好地近似矩阵 $\boldsymbol{R}$，根据更新过的 $\boldsymbol{V}^1$，再次更新 $\boldsymbol{U}^1$，其方法与更新 $\boldsymbol{V}^1$ 类似：

$$\boldsymbol{U}_i^1 = \left[\sum_{j=1}^{M} I_{ij}^R R_{ij}\boldsymbol{V}_j^1\right]\left(\sum_{j=1}^{M} I_{ij}^R(\boldsymbol{V}_j^1)^{\mathrm{T}}(\boldsymbol{V}_j^1) + \lambda_U|\Lambda(\boldsymbol{U}_i)|^{-\beta}\boldsymbol{E}\right)^{-1} \tag{18}$$

(2)当 $n\geqslant 2$ 时。从图 2 可以看出，与单层神经网络最大的不同之处在于，模型当前层的输入数据是上一层的潜在因子矩阵 $\boldsymbol{U}^{n-1}$。其训练过程与单层神经网络类似，具体如下。

首先，初始化潜在因子矩阵 $\boldsymbol{U}^n$。研究后可得：

$$\boldsymbol{U}^n = \begin{bmatrix} u_1^n \\ u_2^n \\ \vdots \\ u_N^n \end{bmatrix} = \begin{bmatrix} h(\boldsymbol{C}_1^n(\boldsymbol{u}_1^{n-1})^{\mathrm{T}} + d_1^n) & \cdots & h(\boldsymbol{C}_D^n(\boldsymbol{u}_1^{n-1})^{\mathrm{T}} + d_D^n) \\ \vdots & \ddots & \vdots \\ h(\boldsymbol{C}_1^n(\boldsymbol{u}_N^{n-1})^{\mathrm{T}} + d_1^n) & \cdots & h(\boldsymbol{C}_D^n(\boldsymbol{u}_N^{n-1})^{\mathrm{T}} + d_D^n) \end{bmatrix} \tag{19}$$

其中，$C_1^n,C_2^n,\cdots,C_D^n$ 表示第 $n$ 层神经网络权重矩阵 $\boldsymbol{C}^n\in\mathbb{R}^{D\times D}$ 中每一行元素；$d_1^n,d_2^n,\cdots,d_D^n$ 表示第 $n$ 层神经网络偏置向量 $d^n\in\mathbb{R}^{1\times D}$ 中每一个元素。

其次，更新潜在因子矩阵 $\boldsymbol{V}^n$。研究后可得：

$$(\boldsymbol{V}_j^n)^{\mathrm{T}} = \left(\sum_{i=1}^{N} I_{ij}^R(\boldsymbol{U}_i^n)^{\mathrm{T}}(\boldsymbol{U}_i^n) + \lambda_V|\Lambda(\boldsymbol{V}_j)|^{-\beta}\boldsymbol{E}\right)^{-1}\left[\sum_{i=1}^{N} I_{ij}^R(\boldsymbol{U}_i^n)^{\mathrm{T}}R_{ij}\right] \tag{20}$$

最后，再次更新潜在因子矩阵 $\boldsymbol{U}^n$。研究后可推得：

$$\boldsymbol{U}_i^n = \left[\sum_{j=1}^{M} I_{ij}^R R_{ij} \boldsymbol{V}_j^n\right] \left(\sum_{j=1}^{M} I_{ij}^R (\boldsymbol{V}_j^n)^{\mathrm{T}}(\boldsymbol{V}_j^n) + \lambda_U |\Lambda(\boldsymbol{U}_i)|^{-\beta}\boldsymbol{E}\right)^{-1} \tag{21}$$

### 2.3　算法设计和复杂度分析

针对 2.2 节中参数更新步骤，研究设计了 IR-NNPMF 模型的算法流程，算法代码详见如下。

**算法　IR-NNPMF 模型算法**

**输入**：已知数据集合 $\Lambda$，$D$，$n$，正则化系数 $\lambda_U$ 和 $\lambda_V$

**输出**：因子矩阵 $\boldsymbol{U}^n$，$\boldsymbol{V}^n$

**Step 1**　当 $n=1$ 时，根据式(15)求解因子矩阵 $\boldsymbol{U}^1$，当 $n \geqslant 2$ 时，根据式(19)求解因子矩阵 $\boldsymbol{U}^n$。

**Step 2**　根据式(20)求解因子矩阵 $\boldsymbol{V}^n$。

**Step 3**　根据式(21)更新因子矩阵 $\boldsymbol{U}^n$。

该算法的主要时间消耗是计算参数更新公式(20)、(21)中的逆矩阵，其时间复杂度为 $\Theta(D^3)$，其中 $D$ 是潜在因子空间的维度。也就是说，$D$ 越小，模型训练所花费的时间越少。然而，$D$ 过小的话，不能很好地表述原来矩阵的特征。因此，在实验过程当中，往往需要取一个合适的 $D$ 值来平衡模型的特征表述和时间复杂度。

此外，神经网络层数 $n$ 的取值也至关重要。$n$ 越大，训练模型所需要的时间就越长；$n$ 过小，模型的预测精度就达不到要求。因此，$n$ 在实验过程当中也需要设置一个合理的取值。

## 3　实验结果与分析

### 3.1　评估指标

本文使用均方根误差(Root Mean Square Error，*RMSE*)和平均绝对误差(Mean Absolute Error，*MAE*)作为实验评估标准[13,21-23]，具体公式为：

$$RMSE = \sqrt{\frac{\sum_{(i,j)} (R_{ij} - \hat{R}_{ij})^2}{|T|}} \tag{22}$$

$$MAE = \frac{\sum_{(i,j)} |R_{ij} - \hat{R}_{ij}|_{abs}}{|T|} \tag{23}$$

其中，$T$ 表示测试集；$|T|$ 表示测试集上已知元素的个数；$R_{ij}$ 表示原始矩阵上的真实值；$\hat{R}_{ij}$ 表示经过模型训练得到的预测值。*RMSE* 和 *MAE* 的值越小，模型的预测精度越高。

### 3.2　实验数据集及预处理

本文选取了 4 个真实工业应用数据集作为实验数据集来验证所提出模型的性能。数据集信息见表 1。

**表 1　数据集信息**

**Tab. 1　Information of datasets**

| 数据集 | 用户 | 项目 | 评分 | 密度/% |
|---|---|---|---|---|
| $D_1$: MovieLens100 k | 943 | 1 682 | 100 000 | 6.300 |
| $D_2$: Jester | 73 421 | 150 | 6 500 000 | 4.200 |
| $D_3$: Amazon-Magazine Subscriptions | 2 428 | 72 098 | 89 689 | 0.050 |
| $D_4$: Amazon-Gift Cards | 1 548 | 128 877 | 147 194 | 0.073 |

为了避免模型训练受到其他因素的影响，本文将所有数据集进行归一化处理；并将所有数据集按照 8:2 划分训练集和测试集，且在训练集上进行 5 折交叉验证；最后进行 10 次独立重复实验，取其平均值作为最终的实验结果。以上所有实验都是用 Pycharm 实现，在 Intel Core i5 CPU 1.40 GHz，8 GB 内存和 AMD Ryzen 7 3700X CPU 3.58 GHz，32 GB 内存的计算机上运行。

### 3.3　实验对比模型设置

为了体现本文提出模型的优越性，这里将与数个模型进行对比。各对比模型信息参见表 2。

**表 2　对比模型信息**

**Tab. 2　Information of compared models**

| 对比模型 | 正则化方法 | 模型训练方法 |
|---|---|---|
| $M_1$: PMF with traditional $L_2$ -regularization | 传统 $L_2$ 正则化 | 随机梯度下降法 |
| $M_2$: PMF with IR | 实例频率加权正则化 | 随机梯度下降法 |
| $M_3$: IR-NNPMF(本文提出模型) | 实例频率加权正则化 | 基于神经网络的方法 |

### 3.4　模型参数设置

在进行对比实验前，需要对模型的参数进行灵敏度测试，并设置最优参数。为了减小模型的复杂度，所有模型中的正则化系数 $\lambda_U$ 和 $\lambda_V$ 设置为 $\lambda_U = \lambda_V = \lambda$。因此，这里需要对以下参数进行训练：正则化系数 $\lambda$，正则化实例频率加权指数 $\beta$，潜在因子维度 $D$，神经网络层数 $n$，具体训练过程如图 3～图 7 所示。

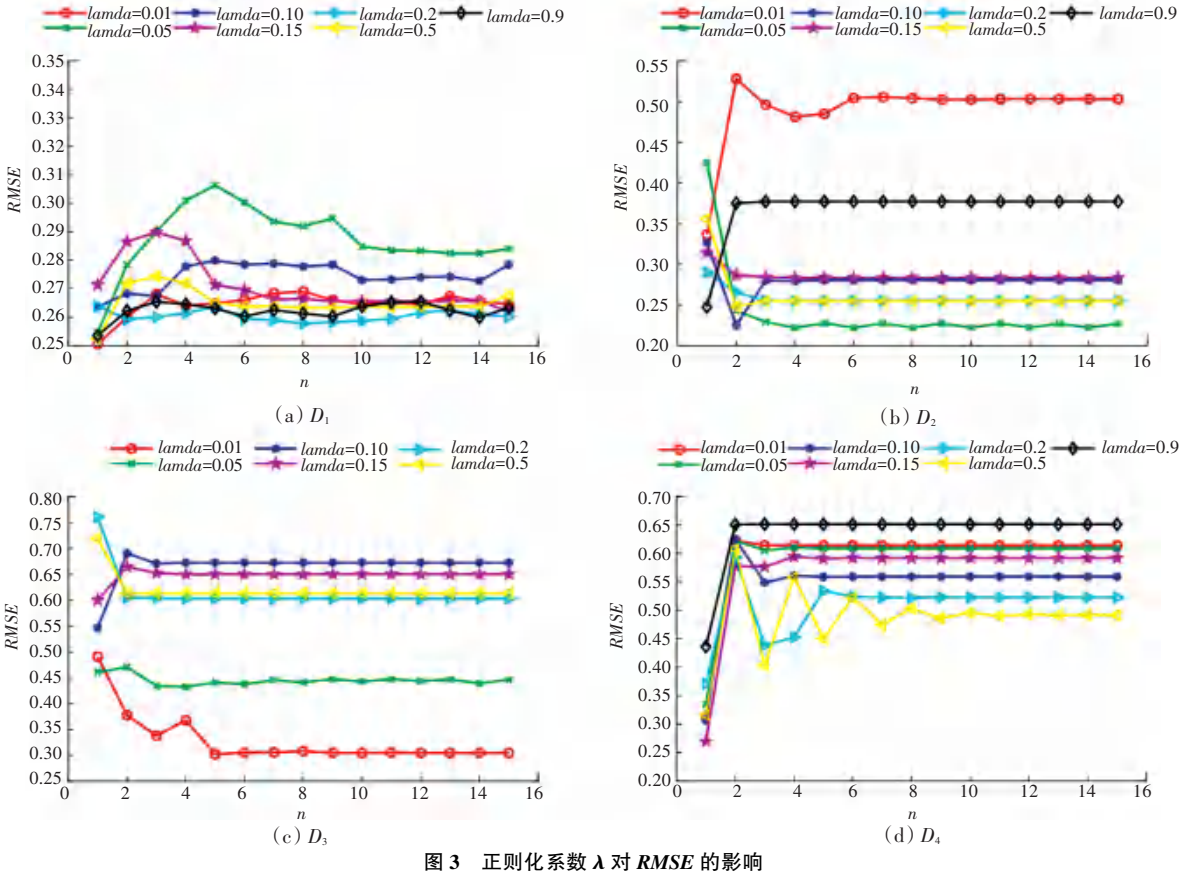


**图 3　正则化系数 $\lambda$ 对 *RMSE* 的影响**

**Fig. 3　Influence of regularization coefficient $\lambda$ on *RMSE***

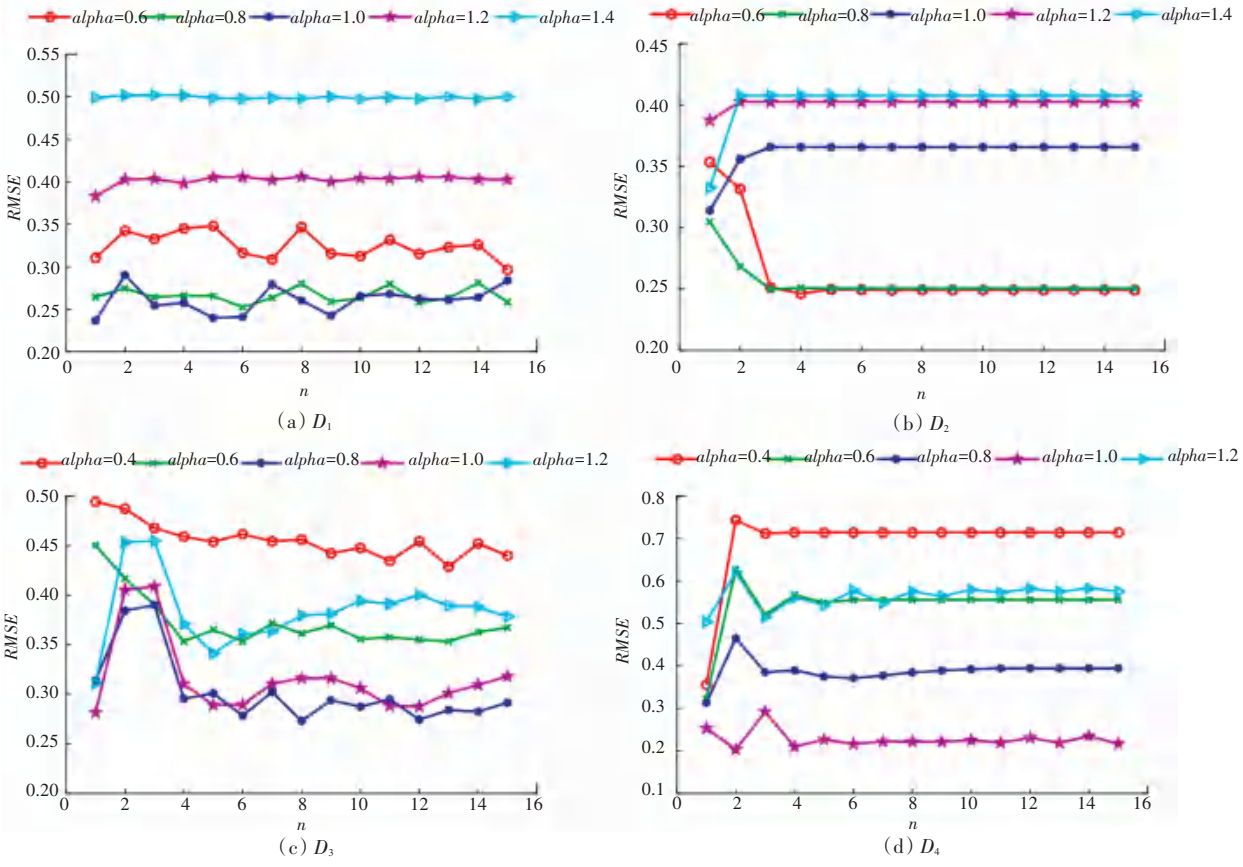


**图 4　正则化实例权重指数 $\beta$ 对 *RMSE* 的影响**

**Fig. 4　Influence of regularization instance-frequency weight index $\beta$ on *RMSE***

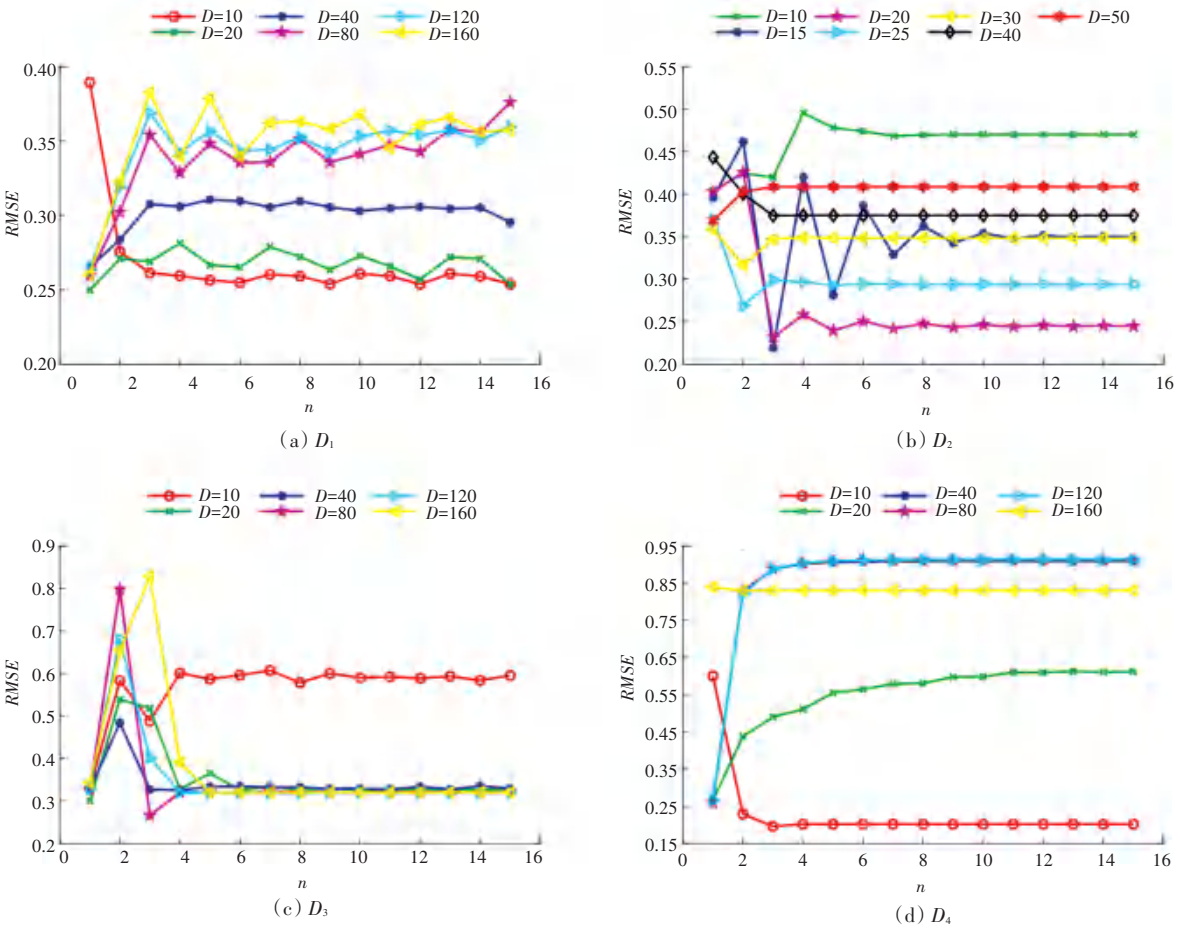


**图 5　潜在因子维数 *D* 对 *RMSE* 的影响**

**Fig. 5　Influence of latent factor dimension *D* on *RMSE***

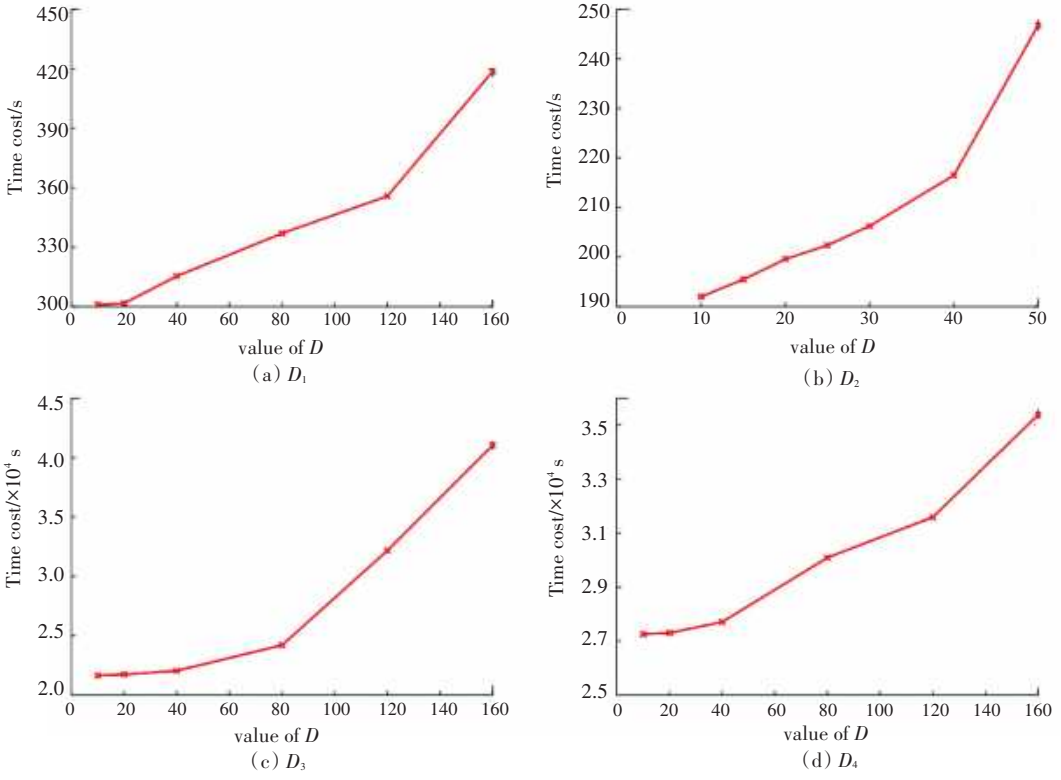


**图 6　不同潜在因子维数 *D* 的时间花费**

**Fig. 6　Time costs with different latent factor dimension *D***

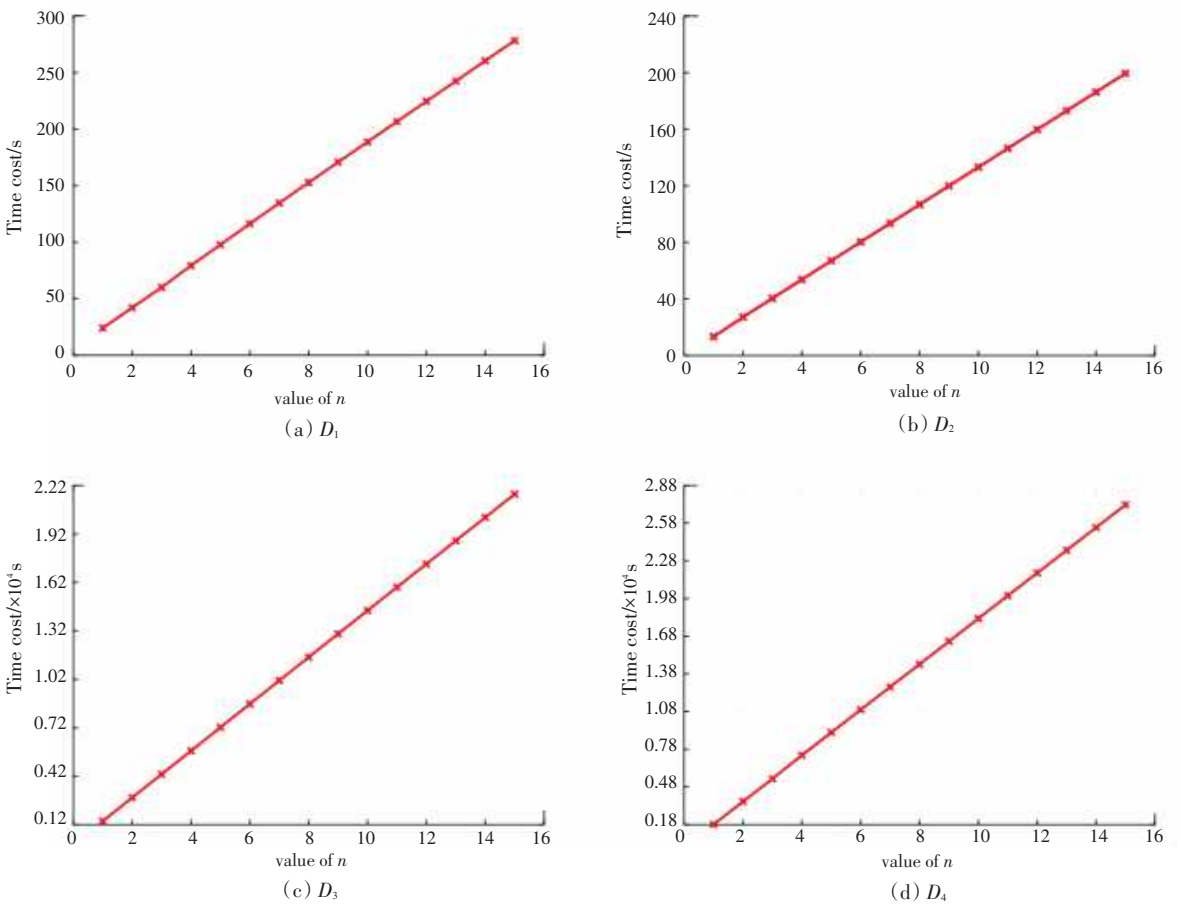


**图 7 不同神经网络层数 $n$ 的时间花费**

**Fig. 7 Time costs with different layers of neural networks**

图 3 展示了正则化系数 $\lambda$ 对模型性能的影响，其中图 3(a)~(d)分别表示模型 $M_3$ 在数据集 $D_1$ ~ $D_4$ 上的参数灵敏度测试结果。可以发现，当正则化实例频率加权指数 $\beta$ 和潜在因子维度 $D$ 固定的情况下，不同数据集的最优正则化参数 $\lambda$ 是不同的。即 $D_1$ ~ $D_4$ 上的最优参数分别为 0.2，0.05，0.01，0.5。

图 4 展示了正则化实例频率加权指数 $\beta$ 对模型性能的影响，其中图 4(a)~(d)分别表示模型 $M_3$ 在数据集 $D_1$ ~ $D_4$ 上的参数灵敏度测试结果。可以发现，当正则化系数 $\lambda$ 和潜在因子维度 $D$ 固定的情况下，$D_1$ ~ $D_4$ 数据集的最优正则化实例权重指数 $\beta$ 分别为 1.0，0.6，0.8，1.0。

图 5 展示了不同神经网络层数下潜在因子维度 $D$ 对模型性能的影响，其中图 5(a)~(d)分别表示模型 $M_3$ 在数据集 $D_1$ ~ $D_4$ 上的参数灵敏度测试结果。可以发现，不同数据集上最优的潜在因子维度 $D$ 是不同的。具体地，在 $D_1$、$D_2$ 和 $D_4$ 上，很容易得出最优的 $D$ 值为 10，20，10；但是，在 $D_3$ 上，$D$ 从 20 增加到 160 对模型精度的影响不明显。当神经网络层数 $n$ = 6 时，和 $D$ = 20 相比，$D$ = 160 时，*RMSE* 仅降低了 0.005 02，考虑到 $D$ 越大，时间复杂度越大，这里选择 $D$ = 20 作为最优参数。

模型参数的设置，除了要对比预测精度外，还要衡量不同参数下的模型训练时间。图 6、图 7 分别展示了在不同潜在因子维度 $D$ 和神经网络层数 $n$ 下的时间消耗。从图 6、图 7 中可以看出，不管在哪个数据集上，随着 $D$ 和 $n$ 的增加，模型的训练时间都会迅速增加。例如，从图 6(c)中，可以看出在数据集 $D_3$ 上，当 $D$ = 20 时，模型的训练时间为 21 742.329 84 s，而 $D$ = 160 时，模型的训练时间为 41 035.049 28 s，增加了 88.73%。虽然随着 $D$ 的增加，模型的表征拟合能力会大大增加，但是所带来的巨大的时间代价，也是难以忍受的。因此选择合适的 $D$ 值是十分重要的。

同样地，从图 7(c)中可以得到，在数据集 $D_3$ 上，当 $n$ = 5 时，训练时间为 7 222.471 204 s，而当 $n$ =15 时，训练时间迅速增加到 21 656.375 45 s。虽然，经过训练的神经网络层数越多，越能更好地拟合原始数据矩阵，但是带来的时间消耗也是巨大的。因此，选择合适的 $n$ 值也是十分必要的。

经过上述分析，研究得到 IR-NNPMF 模型在不同数据集上的参数设置为：在 $D_1$ 上，$\lambda = 0.2, \beta = 0.8, D = 10, n = 2$；在 $D_2$ 上，$\lambda = 0.05, \beta = 0.6, D = 20, n = 4$；在 $D_3$ 上，$\lambda = 0.01, \beta = 0.8, D = 20, n = 6$；在 $D_4$ 上，$\lambda = 0.5, \beta = 1.0, D = 10, n = 4$。

### 3.5　实验结果分析

最后为了评估本文提出模型的有效性，对 3.3 节中所提出的 3 个模型进行对比实验。为了得到一个无偏的结果，$M_1$ 和 $M_2$ 在 4 个数据集上参数设置如下：

对于 $M_2$ 模型，正则化系数 $\lambda$，正则化实例频率加权指数 $\beta$ 和潜在因子维度 $D$，分别与 $M_3$ 模型在各个数据集上的取值相同，并在各个数据集上对学习率 $\alpha$ 在［0.000 1，0.001，0.01，0.05，0.1，0.5，0.9］内进行灵敏度测试，最终数据集 $D_1 \sim D_4$ 的最优参数分别为：0.000 1，0.000 1，0.5，0.1。对于模型 $M_1$，潜在因子维度 $D$ 和学习率 $\alpha$，分别与 $M_2$ 在各个数据集上的取值相同，并在各个数据集上对正则化系数 $\lambda$ 在［0.000 1，0.001，0.01，0.05，0.1，0.5，0.9］内进行参数灵敏度测试，最终数据集 $D_1 \sim D_4$ 的最优参数分别为：0.05，0.05，0.001，0.001。

对比实验的最终结果见表 3。表 3 中，加粗的数据表示该模型在某一指标下为最优值；括号里边的每个数字代表该模型在某一个指标下的排名，其中，1 代表最优，3 代表最差。最终，每一个数据集上的每一个模型都会有一个整体排名，例如，在数据集 $D_2$ 上，模型 $M_1$ 的排名分别为：3、1、3、1，那么其综合排名为：$\frac{3+1+3+1}{4} = 2$。因此，可以得到的结论有：

（1）模型 $M_3$ 在数据集 $D_1 - D_3$ 上关于评价指标 *RMSE* 和 *MAE* 的性能都明显优于模型 $M_1$ 和 $M_2$。例如，在数据集 $D_3$ 上，模型 $M_1$ 和 $M_2$ 取得的最小的 *RMSE* 分别为0.351 85 和0.350 19，而模型 $M_3$ 最小的 *RMSE* 为0.312 59，分别提升了 11.2% 和10.7%。模型 $M_2$ 在数据集 $D_1 \sim D_4$ 上关于评价指标 *RMSE* 的性能均优于模型 $M_1$，例如，在数据集 $D_1$ 上模型 $M_1$ 取得的最小 *RMSE* 为0.345 82，而模型 $M_2$ 取得的最小 *RMSE* 为0.314 97，提升了 8.9%。这说明将数据的不均衡分布信息引入到正则化当中，能有效提高模型的预测精度。

（2）模型 $M_3$ 在所有数据集上的时间消耗都明显小于模型 $M_2$。例如，在数据集 $D_3$ 上，模型 $M_2$ 取得的最小的 *RMSE* 时所花费的时间为4 767.389 6 s；而模型 $M_3$ 所取得最小 *RMSE* 时花费的时间为 2 187.636 5 s。这说明引入神经网络能有效减少模型迭代训练所造成的时间花费。

（3）模型 $M_3$ 在所有数据集上的综合排名明显优于其他对比模型，说明模型 $M_3$ 能够在保证训练时间的同时大大提高预测精度。综上，在同时考虑预测精度和训练时间这 2 个评价指标的情况下，模型 $M_3$ 的性能优于其他对比模型。

**表 3　模型 $M_1 \sim M_3$ 在 $D_1 \sim D_4$ 的最小预测误差和相应时间开销**

**Tab. 3　Lowest prediction error achieved by models $M_1 \sim M_3$ and the corresponding time costs on $D_1 \sim D_4$**

| | $D_1$ | | | | | $D_2$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | *RMSE* | *Time* | *MAE* | *Time* | *Mean Ranking* | *RMSE* | *Time* | *MAE* | *Time* | *Mean Ranking* |
| $M_1$ | 0.345 82(3) | 188.486 2(3) | 0.286 87(3) | 48.276 4(3) | 3 | 0.310 46(3) | **3.143 2(1)** | 0.246 80(3) | **3.143 2(1)** | 2 |
| $M_2$ | 0.314 97(2) | 139.679 1(2) | 0.261 54(2) | 139.679 1(2) | 2 | 0.302 37(2) | 56.829 5(3) | 0.238 12(2) | 50.359 2(3) | 2.5 |
| $M_3$ | **0.240 19(1)** | **43.499 1(1)** | **0.193 61(1)** | **43.499 1(1)** | **1** | **0.243 55(1)** | 14.570 8(2) | **0.187 12(1)** | 14.570 8(2) | **1.5** |
| | $D_3$ | | | | | $D_4$ | | | | |
| | *RMSE* | *Time* | *MAE* | *Time* | *Mean Ranking* | *RMSE* | *Time* | *MAE* | *Time* | *Mean Ranking* |
| $M_1$ | 0.351 85(3) | 2 196.751 7(2) | 0.198 83(2) | 2 196.751 7(2) | 2.25 | 0.199 98(3) | **865.661 76(1)** | 0.096 59(3) | **559.945 0**(1) | 2 |
| $M_2$ | 0.350 19(2) | 4 767.389 6(3) | 0.202 37(3) | 4 767.389 6(3) | 2.75 | 0.195 17(2) | 2 284.482 5(3) | **0.065 83(1)** | 2 284.481 5(3) | 2.25 |
| $M_3$ | **0.312 59(1)** | **2 187.636 5(1)** | **0.156 66(1)** | **2 187.636 5(1)** | **1** | **0.121 47(1)** | 1 491.009 0(2) | 0.093 20(2) | 1 491.008 9(2) | **1.75** |

## 4　结束语

如何有效进行缺失数据的估计一直以来就是数据挖掘领域的研究热点。本文着重考虑模型的预测精度和训练时间两个指标，一方面将数据不均衡信息引入到正则化当中，提高模型的泛化能力；另一方

面,引入神经网络进行模型参数的训练,加快了模型的训练速度,并合理地设置模型的参数,在提高模型预测精度的同时减少模型训练时间。最后,在 4 个真实数据集上的实验结果证明,本文提出的 IR-NNPMF 模型在大型稀疏数据集上的性能是优于其他 PMF 模型的。

## 参考文献


[1] SONG Yan, LI Ming, LUO Xin, et al. Improved symmetric and non-negative matrix factorization models for undirected sparse and large-scaled networks: A trip factorization-based approach [J]. IEEE Transactions on Industrial Informatics, 2020, 16(5): 3006-3017.

[2]刘凯,张立民,周立军. 深度学习在信息推荐系统的应用综述[J]. 小型微型计算机系统, 2019, 40(04): 738-743.

[3] LUO Xin, ZHOU Mengchu, XIA Yunni, et al. Generating highly accurate predictions for missing QoS data via aggregating nonnegative latent factor models [J]. IEEE Transactions on Neural Networks and Learning Systems, 2016, 27(3): 524-537.

[4] LUO Xin, YUAN Ye, ZHOU Mengchu, et al. Nonnegative latent factor model based on β-divergence for recommender systems [J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2019, 51(8): 4612-4623.

[5] ADOMAVICIUS G, TUZHILIN A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions [J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(6): 734-749.

[6] TAKÁCS G, PILÁSZY I, NÉMETH B, et al. Scalable collaborative filtering approaches for large recommender systems [J]. The Journal of Machine Learning Research, 2009, 10: 623-656.

[7] KORENY, BELL R, VOLINSKY C. Matrix - factorization techniques for recommender systems[J] . IEEE Computer, 2009, 42(8): 30-37.

[8] YU Kai, ZHU Shenghuo, LAFFERTY J D, et al. Fast nonparametric matrix factorization for large-scale collaborative-filtering [C] // Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009. Boston, MA, USA: dblp, 2009: 211-218.

[9] MNIH A, SALAKHUTDINOV R. Probabilistic matrix factorization [C]//NIPS. Vancouver, Canada: NIPS Foundation, 2008: 1257-1264.

[10] LUO Xin, ZHOU Mengchu, XIA Yunni, et al. An efficient nonnegative matrix-factorization-based approach to collaborative filtering for recommender systems [J]. IEEE Transactions on Industrial Informatics, 2014, 10(2): 1273-1284.

[11] LUO Xin, SHANG Mingsheng, LI Shuai. Efficient extraction of non-negative latent factors from high-dimensional and sparse matrices in industrial applications[C]// 2016 IEEE 16th International Conference on Data Mining (ICDM). Barcelona, Spain: IEEE, 2016: 311-319.

[12] LUO Xin, ZHOU Mengchu, XIA Yunni, et al. An efficient nonnegative matrix factorization based approach to collaborative filtering for recommender systems [J]. IEEE Transactions on Industrial Informatics, 2014, 10(2): 1273-1284.

[13] LUO Xin, ZHOU Mengchu, LI Shuai, et al. A nonnegative latent factor model for large scale sparse matrices in recommender systems via alternating direction method [J]. IEEE Transactions on Neural Networks and Learning Systems, 2016, 27(3): 579-592.

[14] LUO Xin, LIU Zhigang, LI Shuai, et al. A fast non-negative latent factor model based on generalized momentum method [J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2021, 51(1): 610-620.

[15] SEDHAIN S, MENON A K, SANNER S, et al. AutoRec: autoencoders meet collaborative filtering[C] // Proceedings of the 24th International Conference on World Wide Web. New York, USA: ACM, 2015: 111-112.

[16]冯文,陈志国,傅毅,等. 增强碰撞体算法优化的自编码神经网络[J]. 小型微型计算机系统, 2019, 40(04): 721-725.

[17] JIANG Jiajia, LI Weiling, DONG Ani, et al. A fast deep autoencoder for high-dimensional and sparse matrices in recommender systems [J]. Neurocomputing, 2020, 412: 381-391.

[18] GULIYEV N J, ISMAILOV V E. A single hidden layer feedforward network with only one neuron in the hidden layer can approximate any univariate function [J]. Neural Computation, 2015, 28(7): 1289-1304.

[19] BACCIU D, COLOMBO M, MORELLI D, et al. Randomized neural networks for preference learning with physiological data[J]. Neurocomputing, 2018, 298: 9-20.

[20] YUAN Ye, HE Qiang, LUO Xin, et al. A multilayered-and-randomized latent factor model for high-dimensional and sparse matrices [J]. IEEE Transactions on Big Data, DOI: 10.1109/TBDATA.2020.2988778.

[21]李依桐,黄岳,石川,等. 基于矩阵分解的异质信息网络聚类分析研究[J]. 小型微型计算机系统, 2014, 35(10): 2256-2261.

[22] KOREN Y, BELL R, VOLINSKY C. Matrix factorization techniques for recommender systems [J]. IEEE Computer, 2009, 42(8): 30-37.

[23] MAXWELL F, HARPER F M. The movieLens datasets: History and context [J]. ACM Transactions on Interactive Intelligent Systems, 2015, 5(4): 1-19.