基于深度矩阵分解网络的矩阵填充方法

邝神芬[1,2] 黄业文[3] 宋 杰[1] 李 洽[2]

(韶关学院数学与统计学院 广东 韶关 512005)[1] (中山大学数据科学与计算机学院 广州 510006)[2]

(华南理工大学广州学院计算机工程学院 广州 510800)[3]

摘 要 矩阵分解是矩阵填充中的流行方法,但现有的方法大多是基于浅层的线性模型,当数据矩阵变大且观测数据很少时,容易导致过拟合,性能也随之显著下降。针对这些问题,提出了一种基于深度矩阵分解网络(DMFN)的矩阵填充方法,该方法不仅能弥补传统矩阵分解的缺点,而且能处理复杂的非线性数据。首先,将输入矩阵的观测值对应的行和列向量作为输入,对其进行投影,得到其行(列)的潜在特征向量;然后,分别对行(列)的潜在特征向量构建多层感知器网络;最后,通过构建双线性池化层,将行和列的输出向量进行融合。在推荐系统数据集 MovieLens 及 Netflix 上进行测试,实验结果表明,在相同参数设置下,与主流的填充算法相比,所提方法填充预测的均方误差(RMSE)及绝对值误差(MAE)都有明显提高。

关键词 矩阵填充,矩阵分解,深度学习,双线性池化,多层感知器

中图法分类号 TP18 **文献标识码** A **DOI** 10.11896/jsjkx.190300390

Deep Matrix Factorization Network for Matrix Completion

KUANG Shen-fen[1,2] HUANG Ye-wen[3] SONG Jie[1] LI Qia[2]

(School of Mathematics and Statistics, Shaoguan University, Shaoguan, Guangdong 512005, China)[1]

(School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China)[2]

(School of Computer Engineering, Guangzhou College of South China University of Technology, Guangzhou 510800, China)[3]

Abstract Matrix factorization is a popular technique for matrix completion, but most of the existing methods are based on linear or shallow models, when the data matrix becomes large and the observation data is very small, it is prone to overfitting and the performance decreases significantly. To address this problem, this paper presented a Deep Matrix Factorization Network (DMFN) method, which can not only overcome the shortcoming of traditional matrix factorization, but also deal with complex non-linear data. First, by using rows and columns vectors corresponding to the observed values of the input matrices as input, the latent vector of its row (column) is projected, then the multi-layer perceptron network is constructed on the latent vector of row (column), at last the output vectors of rows and columns are fused by the bilinear pool layer. The proposed algorithm was tested on the standard recommender system dataset (MovieLens and Netflix). Experimental results show that the mean square error (RMSE) and mean absolute error (MAE) of the proposed method are significantly improved compared with the current popular methods under the same parameter setting. The RMSE is 0.830 and MAE is 0.652 on the MovieLens 1M dataset, which increases by about 2% and 6%, respectively. On the Netflix dataset, RMSE is 0.840 and MAE is 0.661, which increases by approximately 1% and 4%, respectively, and achieves optimal results.

Keywords Matrix completion, Matrix factorization, Deep learning, Bilinear pooling, Multilayer perception

1 引言

在科学与工程领域中,数据缺损普遍存在。矩阵填充(Matrix Completion)的目标就是通过对部分已知数据的学习,来对缺损的元素进行预测。矩阵填充已被应用到机器学习、控制理论、计算机视觉等领域,是人工智能领域的核心问题之一。很多问题都可以利用矩阵填充的方法来求解,特别是在推荐系统领域[1-2]。例如,著名的 Netflix 竞赛[1]的任务就是根据用户过去对电影的评分数据来预测用户没有评分的电影,进而根据预测评分的高低把电影推荐给用户。其他热

到稿日期:2019-02-01 返修日期:2019-04-26 本文受韶关学院校级项目(SY2016KJ17,SY2016KJ04),广东省教育厅青年创新人才项目(2018KQNCX233),广东省教育厅基础研究和应用基础研究重点项目(2018KZDXM065),广东省自然科学基金(2017A030307022,2018A0303100015),华南理工大学广州学院优秀骨干教师科研项目(56-CQ18YG18)资助。

邝神芬(1985-),男,博士生,讲师,主要研究方向为数据挖掘与机器学习,E-mail:shfkuang@163.com(通信作者);黄业文(1979-),男,博士生,讲师,主要研究方向为深度学习、数理统计;宋 杰(1974-),男,博士,教授,主要研究方向为数据挖掘、生物信息学与运筹学;李 洽(1985-),男,博士,副教授,主要研究方向为计算优化及其在图像与机器学习中的应用。

门应用包括图像修复[3]、运动结构恢复[4]和图像分类[5]等。在很多应用中,数据大量缺失,如 Netflix 数据集中只有不到1%的用户评分数据,是高度稀疏的。为了使问题适定和有意义,通常假设数据矩阵位于一个非常低的流形上。对于给定的 $m\times n$ 数据缺损矩阵 $\boldsymbol{R}$,若其秩为 k,且 $k\leqslant\min(m,n)$,则可将其分解为 $\boldsymbol{R}=\boldsymbol{U}_{m\times k}\boldsymbol{V}_{k\times n}$。其中,$m$ 表示用户数,n 表示电影数,R_{ij} 表示第 i 个用户对第 j 部电影的评分。将矩阵分解用于推荐系统中主要是基于以下合理的假设:用户对电影的态度或偏好是由少数因素(指标)决定的,低秩结构隐含了每部电影拥有少数的(这里指 k)指标(如电影题材、年龄层次等),V_{ij} 表示电影 j 在指标 i 中的相对分数。另外,U_{ij} 表示第 i 个用户对第 j 个指标的偏好程度。当已知的观测数据的位置在矩阵中是随机的,且满足 $k\ll\frac{m\times n}{m+n}$ 时,则对数据矩阵 WDT 分解后的未知参数有 $(m+n)\times k$ 个,这远远小于 $m\times n$ 个。因此,理论上只需要很少的观测值即可求出 $\boldsymbol{U},\boldsymbol{V}$ 的解,进而可以通过用户与电影的潜在特征向量的内积来求得缺损数据的预测值。其原理如图 1 所示,对于缺损的电影评分矩阵 $\boldsymbol{R}$,矩阵分解的本质就是通过部分已知观测值来求出用户和电影的潜在特征向量或潜向量(*Latent Vector*),即将 $\boldsymbol{R}$ 分解为两个潜在特征矩阵 $\boldsymbol{U}$ 和 $\boldsymbol{V}$ 的乘积。从而,第 i 个用户用潜在特征向量 $\boldsymbol{U}^{(i)}$ 表示,即 $\boldsymbol{U}$ 的第 i 行;第 j 部电影可由潜在特征向量 $\boldsymbol{V}_j$ 表示,即 $\boldsymbol{V}$ 的第 j 列。对于缺损的数据 R_{ij},即第 i 个用户对第 j 部电影的评分(如图 1 中的黑格子的位置),可通过其对应行列的潜在特征向量的内积来进行预测,可表示为 $\hat{r}_{ij}=\langle\boldsymbol{U}^{(i)^{\mathrm{T}}},\boldsymbol{V}_j\rangle$。

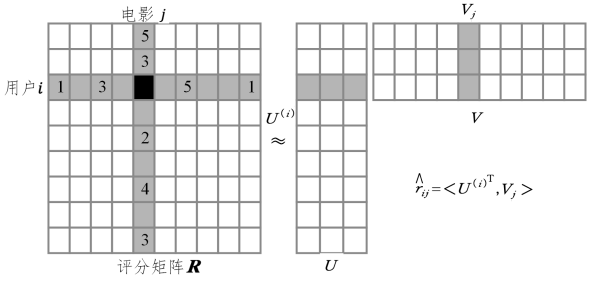

图 1 将电影评分矩阵分解为两个潜在特征矩阵的乘积

Fig. 1 Decomposing the movies rating matrix as multiplication of two latent matrices

在矩阵填充算法中,最常用的方法是矩阵分解及其改进方法,如文献[1,6-8]中的方法,这得益于其在推荐系统(特别是 Netflix 竞赛)中的成功应用。但是,这些方法都是基于浅层的模型,本质上是对数据矩阵的行(用户)和列(电影)的线性关系进行建模。这些模型都假设缺损的数据矩阵是低秩的,且可线性地由潜在特征向量生成,即对缺损数据的预测是直接通过潜在特征向量的线性交互得到的。但当缺损数据变得复杂而多样,如已知数据集不是呈某种随机分布或已知观测数据的数量太少时[9],其效果往往不理想。近年来,随着计算机计算能力的提升、数据量的不断增大,以及深度学习[10-11]在图像分类领域的突破,很多学者将现有的深度学习方法应用到矩阵填充中,特别是应用到基于协同过滤的推荐系统[2,12]上,如协同过滤-受限波兹曼机(RBM-CF)[13]、自编码器(Auto Encoder)[14]、变分自编码器[15]、递归神经网络[16]、卷积网络[17]、多层感知器[18]等,获得了优于浅层模型的结果。最近,Zhang 等[19]对这方面的研究做了全面和系统的综述。现阶段,基于深度学习的矩阵填充算法一般包含多个网络层,如输入层、嵌入层(Embedding)、非线性变换层(如 Sigmoid 和 Relu)、全连接层和输出层等,对构建的网络一般利用随机梯度下降法进行参数的反向传播训练。与浅层矩阵分解不同的是,基于深度学习的矩阵分解一般通过对潜在特征向量做多层非线性映射得到模型的预测值。

将深度学习技术应用到矩阵分解中,一般是将数据矩阵中元素对应的行和列分别嵌入到潜向量空间。由于数据通常位于低维流形上,因此嵌入向量的维数一般比数据矩阵的阶小得多。浅层矩阵分解一般直接利用潜向量来构造线性模型。这些方法由于直接对潜向量做线性交互,因此要想得到较好的恢复结果,需要对已知数据的特征空间附加很多条件,如文献[20]所述,对已知的观测数据的数量和概率分布都是有要求,但在实际应用中很难满足这些条件。对此,文献[21]提出对潜在变量做非线性变换,使得其能够减少这些限制,并且能够将副信息(Side Information)包含到模型中。如在 Netflix 竞赛数据集中,除了电影评分外,还有很多额外的副信息,如用户的基本信息、电影的类别等,现有的基于线性模型的方法很难利用这些信息。但是该非线性变换也存在不少问题,如果副信息不完整或者根本没有副信息,那么该方法将受到诸多限制。深度学习由于能够学习到数据隐藏的复杂的非线性结构特征,因此成为了人工智能领域的核心技术之一。基于深度学习的矩阵分解算法一般对潜向量进行非线性变换,通过构造多层神经网络来进行训练。网络的输入一般是潜在特征向量,输出是该元素的模型预测值。例如,文献[22]提出的深度矩阵填充算法,在构建的网络中网络参数与输入的潜向量都是作为未知参数进行学习的,通过已知元素的实际值与模型得到的值来构建损失函数,对隐变量和网络参数同时进行优化,通过随机梯度下降法求得潜在变量与网络参数的数值解。

本文主要结合深度学习与矩阵分解,提出了一种深度矩阵分解网络(Deep Matrix Factorization Network,DMFN),该方法用多层感知器将潜在特征向量扩展到多层的神经网络。作为对文献[18,22-23]中的方法的改进,本文主要有以下两个创新点:

1)提出了一种基于深度矩阵分解网络的矩阵填充框架,该框架将传统的矩阵分解模型扩展到多层神经网络,不仅能弥补传统矩阵分解的不足,还能处理复杂的非线性数据;

2)同时利用矩阵的行和列构造多层网络,与传统网络直接对输出进行内积不同,本文构造双线性池化层,利用 Hadamard 乘积,对矩阵行和列潜在特征向量的多层感知器输出,通过双线性映射来进行融合,从而得到模型的预测值。

[1)] https://github.com/shfkuang/dmfn

2　相关工作

一般地，低秩矩阵填充可以被抽象为以下数学问题：

$$\min_{\boldsymbol{Y}} rank(\boldsymbol{Y})$$

$$\text{s.t. } R_{ij}=Y_{ij},\ (i,j)\in\Omega$$

其中，$\Omega\subseteq\{1,2,\cdots,m\}\times\{1,2,\cdots,n\}$是已知的观测值在矩阵中的索引的集合，$rank(\boldsymbol{Y})$表示矩阵$\boldsymbol{Y}$的秩。显然，直接求解该问题是 NP 难的。通常的方法是利用低秩约束来求得数值解，而近年来已经有大量的学者尝试用不同方法来解决该问题。这些方法可以粗略地归为两类：核正则化(Spectral Regularization)与矩阵分解。核正则化主要通过寻找一个秩函数的代理来约束解的低秩性。近年来，随着压缩感知理论[20]的突破，利用核范数(Nuclear Norm)[24]求解低秩解不仅在理论上可靠，在实践中也非常有效。其通常被转化为以下的拉格朗日形式进行求解：

$$\min_{\boldsymbol{Y}}\frac{1}{2}\sum_{(i,j)\in\Omega}(R_{ij}-Y_{ij})^2+\lambda\|\boldsymbol{Y}\|_* \tag{1}$$

其中，$\|\boldsymbol{Y}\|_*$是矩阵$\boldsymbol{Y}$的核范数，定义为其特征值之和。问题(1)是凸的，一种经典的求解算法是利用奇异值阈值算法(Singular Value Threshold)[25]迭代求出最优解。文献[24]证明了核范数在单元球上是最紧凸松弛代理；文献[20]进一步证明了若数据有低秩的结构，则在某种条件下，求解式(1)能精确恢复出矩阵$\boldsymbol{X}$。然而，在实际应用中，由于求解问题(1)涉及到奇异值分解(SVD)，其计算量随着矩阵变大或者秩增加而变得庞大。另外，为了求得低秩解，核范数求解过程中每次迭代都需要对全部特征值进行同等的收缩，即所有特征值都减去相同的阈值，这会导致过分收缩某些包含重要信息的特征值，从而丢失重要信息[26]。一种改进的方法是用非凸正则化[3,26]来近似矩阵秩，其在实际应用中能取得比核范数更好的效果；但是该方法同样涉及 SVD 分解，在大规模数据上的应用也因此受到限制。

矩阵分解是另一种主要的矩阵填充方法，即直接将矩阵分解为两个(或更多)低秩矩阵的乘积，这样就能显式地满足低秩约束。标准的矩阵分解一般是求解以下问题：

$$\min_{\boldsymbol{U},\boldsymbol{V}}\frac{1}{2}\sum_{(i,j)\in\Omega}(R_{ij}-\boldsymbol{U}^{(i)}\boldsymbol{V}_j)^2+\lambda(\|\boldsymbol{U}\|_F^2+\|\boldsymbol{V}\|_F^2) \tag{2}$$

该方法也被称为最大边缘矩阵分解(MMMF)[27]。式(2)的形式也可以通过式(1)来导出。矩阵分解的优点是计算速度快，内存需求少，可通过交替优化$\boldsymbol{U},\boldsymbol{V}$来求解。当数据噪声是高斯噪声时，文献[8]提出概率矩阵分解算法，通过最大似然估计进行求解，其导出的结果与式(2)具有相同的形式。由于没有对变量做约束，直接进行线性分解将导致其解不唯一。因此，文献[28]通过将式(2)的损失函数改为 L_1 范数，使用 MM(Majorization Minimization)方法进行求解，该方法在很多视觉应用中获得了很好的效果。文献[29]通过对变量施加正交约束，并利用黎曼优化来进行求解。由于正交约束减小了解的搜索空间，该方法的迭代收敛速度比交替优化快。但这些方法本质上都是基于浅层线性模型的，隐性地假设数据能通过矩阵行和列的潜向量线性生成，因此其实用性受到很多限制。

最近，研究者集中于结合深度学习和矩阵分解来提高矩阵填充的效果。文献[23]提出基于矩阵分解的神经网络，其主要思想是将数据矩阵的行和列的潜向量构造成一个多层的神经网络，并交替优化潜在变量和网络参数，尽管效果并没有比浅层模型有太大提升，但是其思想是很多深层矩阵填充算法的基础。文献[30]基于同样的原理，提出用深层非负半监督矩阵分解(Deep Semi-NMF)来学习数据的复杂的层次信息。文献[30]基于非线性的潜向量来学习多层矩阵分解，其本质是通过多层非线性映射将高维数据矩阵分解为低维矩阵的乘积。文献[31]提出用多层感知器将潜在变量扩展到多层网络中，并直接通过余弦相似度来融合多层感知器网络中的矩阵行和列向量的输出，进而得到模型的输出值，在 TOP-K 推荐系统中获得了令人满意的结果。与文献[31]不同，本文提出的 DMFN 在多层感知器层后并不直接输出模型值，而是再训练一个双线性模型，这样能更好地融合行和列的潜在特征向量。文献[32]提出了神经元协同过滤模型，该方法同时训练两个网络，一个是多层感知器，另一个是矩阵分解，最后将它们组合起来。在推荐系统的某些数据集测试中，该算法获得了优异的性能，但该方法只是在最后模型预测过程中简单地对多层感知器和矩阵分解做连接。而本文提出的算法则直接在矩阵分解中嵌入多层感知器来扩展网络，使得多层感知器与矩阵分解无缝连接，在统一的模型下进行训练。

3　深度矩阵分解网络

深度矩阵分解网络(DMFN)主要通过组合不同的层来提高非线性处理的能力。本文提出的算法如图 2 所示，主要包含输入层、嵌入层、多层感知器层、双线性池化层和输出层。

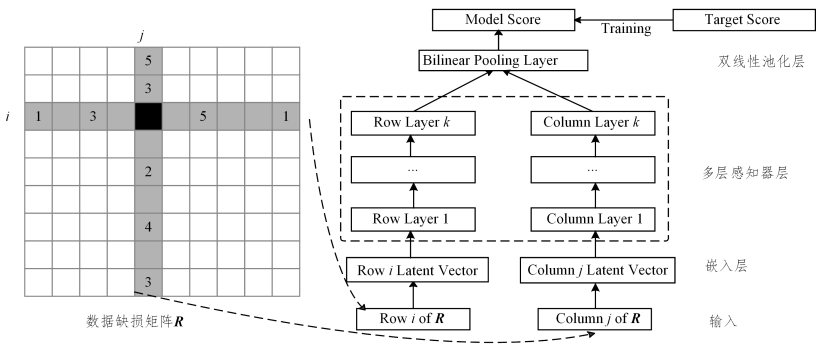

图 2　深度矩阵分解网络的结构

Fig. 2　Structure of deep matrix factorization network

在训练阶段,对于每个观测点,通过其对应的行和列向量进行变换来得到潜在特征向量,然后将其作为多层感知器层的输入,并通过双线性池化层对行列潜在特征向量进行融合,最后利用模型值与已知的观测值构建平方损失函数。

3.1 输入层

数据矩阵中,每个元素以〈row_id,col_id,rating〉的格式输入。在训练阶段,对于矩阵中的所有已知元素(设有 d 个),其输入为 $d\times 3$ 的矩阵,第一列为该元素所在矩阵行的索引,第二列为该元素所在矩阵列的索引,第三列为对应的评分值。在测试阶段,对于所有未知元素(设有 k 个),输入为 $k\times 2$,每行对应〈row_id,col_id〉,其 rating 未知,矩阵填充就是通过已知观测值去训练模型,从而预测这 k 个未知元素对应的 rating 值。

3.2 嵌入层

嵌入层主要是将输入层映射到矩阵行(列)嵌入矩阵。对于一个 $m\times n$ 的缺损矩阵 $\boldsymbol{R}$,主要是对其中的每个元素 R_{ij} 构造输入向量。本文直接把观测值在矩阵中对应的第 i 行和第 j 列向量作为输入,其缺损元素用 0 表示。即对于观测值 R_{ij},其输入是 $\boldsymbol{R}^{(i)}$ 和 $\boldsymbol{R}_j$,分别对应矩阵 $\boldsymbol{R}$ 的第 i 行和第 j 列。在训练阶段,主要是对所有 $(i,j)\in\Omega$,以其对应行和列向量作为输入,并通过图 2 所示的网络进行映射,从而得到输出值,然后利用输出的模型值与观测值构造损失函数。直接将矩阵中元素对应的行和列向量作为该观测值的输入,其信息是冗余的。为了得到其潜在的特征向量,这里对输入的行和列向量分别做映射,将其投影到低维的空间。假设潜在特征向量的维数是 k(k 远远小于数据矩阵的阶),则分别通过以下非线性变换将输入投影到潜在特征向量。对于给定的 R_{ij},有:

$$\boldsymbol{x}_u=\sigma(\boldsymbol{R}^{(i)}\boldsymbol{W}_u)$$

$$\boldsymbol{x}_v=\sigma(\boldsymbol{W}_v\boldsymbol{R}_j)$$

其中,$\boldsymbol{x}_u^{\mathrm{T}},\boldsymbol{x}_v\in\boldsymbol{R}^k$ 分别为非线性变换得到的行和列的潜在特征向量;$\boldsymbol{W}_u\in\boldsymbol{R}^{m\times k},\boldsymbol{W}_v\in\boldsymbol{R}^{k\times n}$ 分别为矩阵行和列向量的特征投影矩阵,是未知的训练参数,需要通过反向传播来训练;$\sigma(x)$ 是非线性激活函数,主要是对线性变换加入非线性因素,以弥补线性模型的不足,这是深度分解网络与传统矩阵分解的区别,也是深度学习成功的重要因素。常见的激活函数有 Sigmoid,ReLu,Tanh 等。然而,在矩阵分解中,Sigmoid 的效果通常比其他激活函数的效果好,文献[14]中的实验也有类似结论,本文实验部分将验证这一结论。由于激活函数不改变向量维数,因此对于输入为 $d\times 3$ 的矩阵,其行和列嵌入矩阵维数均为 $d\times k$,其中每行向量对应其行(列)潜在特征向量。

3.3 多层感知器层

多层感知器(Multi-layer Perception,MLP)主要是将潜在特征向量进行多层的非线性映射,从而更好地学习数据的非线性特征。该层主要通过多层感知器将潜在特征向量进行全连接。假设矩阵的行和列潜在特征向量分别为 $\boldsymbol{x}_u$ 和 $\boldsymbol{x}_v$,第 i 层的行和列向量的感知器权重矩阵和偏差向量分别为 $\boldsymbol{W}_i'\in\boldsymbol{R}^{k\times k},\boldsymbol{b}_i'\in\boldsymbol{R}^k$ 及 $\boldsymbol{W}_i\in\boldsymbol{R}^{k\times k},\boldsymbol{b}_i\in\boldsymbol{R}^k,i=1,\cdots,L$,其中 L 表示网络层的深度;并用 $\boldsymbol{H}_i(i=1,\cdots,L-1)$ 表示中间的隐藏层,每个隐藏层都有与潜在特征向量相同的单元。若行和列向量的多层感知器的输出分别用 $\boldsymbol{p}_u$ 和 $\boldsymbol{q}_v$ 表示,则行潜在特征向量的输出可表示为:

$$\boldsymbol{H}_0=\boldsymbol{x}_u$$

$$\boldsymbol{H}_i=\sigma(\boldsymbol{H}_{i-1}\boldsymbol{W}_i'+\boldsymbol{b}_i'),i=1,\cdots,L-1$$

$$\boldsymbol{p}_u=\sigma(\boldsymbol{H}_{L-1}\boldsymbol{W}_L'+\boldsymbol{b}_L')$$

而对于列潜在特征向量,则由以下映射生成:

$$\boldsymbol{H}_0=\boldsymbol{x}_v$$

$$\boldsymbol{H}_i=\sigma(\boldsymbol{W}_i\boldsymbol{H}_{i-1}+\boldsymbol{b}_i),i=1,\cdots,L-1$$

$$\boldsymbol{q}_v=\sigma(\boldsymbol{W}_L\boldsymbol{H}_{L-1}+\boldsymbol{b}_L)$$

将以上公式简记为 $\mathrm{MLP}_r(\boldsymbol{x}_u)=\boldsymbol{p}_u$ 和 $\mathrm{MLP}_c(\boldsymbol{x}_v)=\boldsymbol{q}_v$。由于多层感知器本质上是对输入做全连接,且本文定义为等维度的非线性映射,因此层数一般不宜太高,否则容易过拟合。

3.4 双线性池化层

对矩阵行和列的潜在特征向量进行多层感知器映射后,融合行和列的网络输出以得到最后的模型值,通常的方法是直接进行内积[15]。为了进一步提高泛化与学习能力,本文构建双线性池化层(Bilinear Pooling),该方法在问答系统和细粒度图像分类中有广泛应用。本文讨论以下几种不同形式的双线性池化。

对于给定的输入 $\boldsymbol{x}\in\boldsymbol{R}^k$ 和 $\boldsymbol{y}\in\boldsymbol{R}^k$,传统方法是直接进行内积。与此不同的是,双线性模型主要对这两个输入向量做以下的变换:

$$m=\boldsymbol{x}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{y}=\sum_{i=1}^{k}\sum_{j=1}^{k}x_i w_{ij}y_j \tag{3}$$

其中,$\boldsymbol{W}\in\boldsymbol{R}^{k\times k}$ 是一个权重矩阵,是引入的未知参数。通常,因为在网络中对矩阵 $\boldsymbol{W}$ 施加正则化以防止拟合,所以 $\boldsymbol{W}$ 一般是稀疏的。因此,可将 $\boldsymbol{W}$ 做低秩分解,即 $\boldsymbol{W}=\boldsymbol{U}\boldsymbol{V}$,这里 $\boldsymbol{U}\in\boldsymbol{R}^{k\times t},\boldsymbol{V}\in\boldsymbol{R}^{t\times k}$,$t\ll k$,则式(3)可化简为:

$$m=(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{U})(\boldsymbol{V}\boldsymbol{y})=\boldsymbol{1}^{\mathrm{T}}(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x}\odot\boldsymbol{V}\boldsymbol{y})$$

其中,$\boldsymbol{1}^{\mathrm{T}}$ 是一个长为 k、值全为 1 的行向量;符号 $\odot$ 表示 Hadamard 乘积。为了进一步提高其非线性建模能力,将长为 k 的行向量用一个 $\boldsymbol{h}\in\boldsymbol{R}^k$ 的参数向量来表示,从而上式可表示为:

$$m=\boldsymbol{h}^{\mathrm{T}}(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x}\odot\boldsymbol{V}\boldsymbol{y})$$

对上式的 Hadamard 乘积再施加激活函数,这样就得到了低秩双线性池化层。根据对施加激活函数位置的不同,可有以下两种不同的表示:

$$m=\boldsymbol{h}^{\mathrm{T}}\{\sigma(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x})\odot\sigma(\boldsymbol{V}\boldsymbol{y})\} \tag{4}$$

$$m=\boldsymbol{h}^{\mathrm{T}}\sigma(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x}\odot\boldsymbol{V}\boldsymbol{y}) \tag{5}$$

其中,$\boldsymbol{h}$ 是需要训练的未知参数。由于 $\sigma(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x})$ 及 $\sigma(\boldsymbol{V}\boldsymbol{y})$ 与多层感知器有相同的形式,因此式(5)可被进一步简化。可将 $\boldsymbol{U},\boldsymbol{V}$ 看作多层感知器的训练参数,若多层感知器层的行和列潜在特征向量输出分别为 $\boldsymbol{p}_u$ 和 $\boldsymbol{q}_v$,则有:

$$m=\boldsymbol{h}^{\mathrm{T}}\sigma(\boldsymbol{p}_u^{\mathrm{T}}\odot\boldsymbol{q}_v) \tag{6}$$

该式可以看成是加权形式的内积。

3.5 损失函数

最后讨论两种损失函数:单层的非线性函数和利用多层

感知器来构造基于深度学习的损失函数。

3.5.1　单层网络损失函数

在单层网络的情况下，对输入向量做变换得到潜在特征向量后，直接利用双线性池化层对行和列的潜在特征向量进行融合来得到模型的预测值。可构建如下的损失函数：

$$\min_{\boldsymbol{W}_u,\boldsymbol{W}_v,\boldsymbol{W}} \frac{1}{2}\sum_{(i,j)\in\Omega}(R_{ij}-\phi_r(\boldsymbol{R}^{(i)})\boldsymbol{W}\phi_c(\boldsymbol{R}_j))^2+\lambda(\|\boldsymbol{W}_u\|_F^2+\|\boldsymbol{W}_v\|_F^2+\|\boldsymbol{W}\|_F^2) \tag{7}$$

其中，$\phi_r(\boldsymbol{R}^{(i)})=\sigma(\boldsymbol{R}^{(i)}\boldsymbol{W}_u)$，$\phi_c(\boldsymbol{R}_j)=\sigma(\boldsymbol{W}_v\boldsymbol{R}_j)$。可以看出，式(7)和文献[21]有着相同的形式，但不同的是，文献[21]通过核映射将其映射到高维空间，而式(7)则直接对行列向量做非线性映射。

3.5.2　多层网络损失函数

对于有多层感知器的网络，损失函数可抽象地表示为如下非线性问题：

$$\min_{\boldsymbol{W}_u,\boldsymbol{W}_v,\boldsymbol{W}} \frac{1}{2}\sum_{(i,j)\in\Omega}(R_{ij}-\psi(\boldsymbol{R}^{(i)},\boldsymbol{R}_j|\boldsymbol{W}))^2+\lambda\|\boldsymbol{W}\|$$

其中，$\boldsymbol{W}$ 是网络中的未知参数集合；$\psi(\cdot|\boldsymbol{W})$ 是网络中以矩阵行和列向量作为输入的多层非线性映射，其输出是网络的模型预测值；$\|\boldsymbol{W}\|$ 是对参数进行正则化，这里采用 L_2 范数来防止过拟合。考虑有 k 层感知器的情况，并使用式(6)进行池化，那么损失函数可进一步表示为：

$$\min_{\boldsymbol{W},\boldsymbol{W}_u,\boldsymbol{W}_v,h} \frac{1}{2}\sum_{(i,j)\in\Omega}(R_{ij}-\boldsymbol{h}^{\mathrm{T}}\sigma(\boldsymbol{p}_{\boldsymbol{U}^{(i)}}^{\mathrm{T}}\odot\boldsymbol{q}_{\boldsymbol{V}_j}))^2+\lambda(\|\boldsymbol{W}_u\|_F^2+\|\boldsymbol{W}_v\|_F^2+\|\boldsymbol{W}\|)$$

其中，$\boldsymbol{p}_{\boldsymbol{U}^{(i)}}=\mathrm{MLP}_r(\boldsymbol{R}^{(i)})$ 和 $\boldsymbol{q}_{\boldsymbol{V}_j}=\mathrm{MLP}_c(\boldsymbol{R}_j)$ 是根据 3.3 节讨论的多层感知器计算得到的输出值；$W=\{\boldsymbol{W}_i,\boldsymbol{b}_i,\boldsymbol{W}_i',\boldsymbol{b}_i'\}$ $(i=1,2,\cdots,k)$ 为网络参数集合；而 $\|\boldsymbol{W}\|$ 则表示分别对该集合中的参数做正则化约束。

4　实验

4.1　实验配置

为了评估所提算法的性能，将其与经典矩阵分解算法及目前主流的深度学习算法进行比较。参与比较的算法主要包括以下 4 种：RBM-CF[13]，BiasMF[1]，NNMF[23] 以及 AutoRec[14]。RBM 和 AutoRec 统一采用矩阵列作为输入，即仅比较(I-RBM-CF)及(I-AutoRec)。算法运行的环境为：Ubuntu 16.04 系统，Intel Core i7 CPU，32 GB 内存及 NVidia 1080TI 11G 显卡。本文采用推荐系统的标准数据集来比较算法性能，包括 MovieLens 100 k，1 M 及 10 M 数据集(下文分别用 ml-100 k，ml-1m 和 ml-10m 表示)，以及 Netflix 数据集。实验中只考虑用户对项目的评分，而不考虑其他副信息。4 个评估数据集的基本信息如表 1 如列。Netflix 数据集现在由 Kaggle 平台管理，由于官方没有提供其 Qualify 集的真实值，因此需在训练集中随机选取部分数据作为测试集，而 Probe 集依然作为验证集。实验对所有训练的数据集仅提取如下信息：用户 ID、电影 ID 及评分。评分均为 1 到 5 的整数；而用户 ID 和电影 ID 是从 1 开始计数，并且数字可能是不连续的。本文提出的算法采用 Tensorflow 1.8 来实现，训练时采用 GPU，实验代码可在作者的 GitHub 主页[1)] 下载。

表 1　推荐系统数据集的基本信息

Table 1　Basic information of recommender system datasets

数据集	用户数	项目数	评分数	稀疏度
ml-100 k	944	1 683	100 000	0.062 9
ml-1 m	6 040	3 706	1 000 209	0.044 7
ml-10 m	6 040	3 706	1 000 209	0.044 7
Netflix	480 189	17 770	100 480 507	0.011 8

4.2　评估标准

为了评估算法的性能，将 MovieLens 数据集随机进行分割，90%用于训练(训练集中再取 2%作为验证数据集)，剩下的 10%用于测试。对于 Netflix 数据集，随机在训练集中选取与 Qualify 集数量相同的数据集作为测试集。采用两个常用的评估准则来检验算法的性能，即均方误差 (RMSE)和均方绝对值误差(MAE)。

$$RMSE=\left(\frac{\sum_{(i,j)\in\Omega}(X_{ij}-\hat{X}_{ij})^2}{|\Omega|}\right)^{\frac{1}{2}}$$

$$MAE=\frac{\sum_{(i,j)\in\Omega}|X_{ij}-\hat{X}_{ij}|}{|\Omega|}$$

其中，Ω 是可观测到的数据的索引，而 $|\Omega|$ 表示观测数据的个数。

4.3　参数设置

与其他深度学习算法一样，深度分解网络涉及大量要训练的参数，在训练前需要将其初始化，本文用标准正态分布来初始化所有训练参数。正则化参数设置为 $\lambda=0.5$，并采用 Adam 优化器做随机梯度下降训练，其批量大小(batch size)设置为 25 000，学习率设置为 0.000 1，其他参数采用 Tensorflow 的默认值。实验中使用 Sigmoid 作为激活函数，并且使用式(6)进行双线性池化。实验中，行和列嵌入维数即潜在特征向量均取相同维数，对 MovieLens 100 k，1 m，10 m 及 Netflix 数据集分别取行(列)维数为 300，500，1 000，1 000。所有算法均不采用预训练模型，且均使用 GPU 进行训练。

4.4　实验结果

实验 1　算法收敛性分析

首先，在 ml-100 k 和 ml-1 m 上对算法的收敛性进行测试。对于 ml-100k，采用 1 层的感知器网络；对于而 ml-1m，采用 2 层的感知器网络。两种数据集上损失函数的变化情况如图 3 和图 4 所示。在这两个数据集上，损失函数随着训练周期(epoch)的增加而迅速下降，ml-100 k 在 $epoch<50$ 时目标函数呈指数下降；但随着训练周期的增加，其目标函数下降速度变慢，最后目标函数基本不再减小，即达到收敛。而在 ml-1 m 数据集上，当 $epoch<500$ 时，目标函数单调递减；当 $epoch>500$ 时，目标函数基本不降低。由于学习率设置较小时，两个数据集都需要大量的迭代，因此将学习率设置为较低的 0.000 1，以保证迭代结果的稳定性。如果把学习率设置为 0.01，则 ml-100 k 在 $epoch=100$ 时收敛，而 ml-1m 在 $epoch=70$ 时收敛，但其结果都存在一定的抖动。

1) http://github.com/shfkuang/dmfn

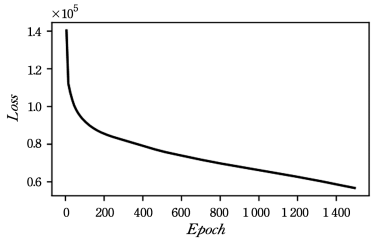

图 3 ml-100 k 上损失函数的变化情况

Fig. 3 Loss function changes over Epoch on ml-100 k dataset

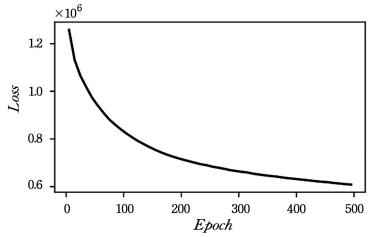

图 4 ml-1 m 上损失函数的变化情况

Fig. 4 Loss function changes over Epoch on ml-1 m dataset

图 5 和图 6 主要给出了训练过程中测试误差(RMSE 和 MAE)随着训练周期(epoch)增加的变化情况。对于 ml-100 k,当训练周期增加至 150 次后,其误差开始缓慢减小;当 epoch 次数达到 500 时,$RMSE=0.889$,$MAE=0.697$。而对于 ml-1m 数据集,当 epoch 达到 70 时,误差开始缓慢减小;当 epoch=500 时,$RMSE=0.837$,$MAE=0.655$。从图 6 中可以发现,测试结果存在一定的抖动,这主要是由于采用随机梯度下降导致的。由于观测数据多于 80 万,而每次只取 2.5 万的小批量数据来训练,因此测试结果有一定的波动。由图 3－图 6 可以看到,测试误差整体上是随着训练误差的减小而减小,这证明了算法是有效的,其误差随着迭代次数的增加而收敛到稳定值。

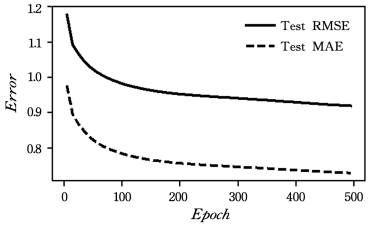

图 5 ml-100k 测试误差

Fig. 5 Test error of ml-100 k dataset

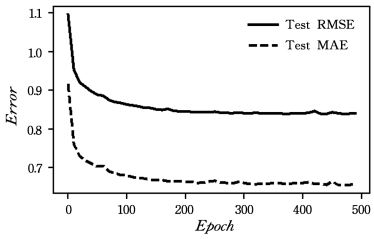

图 6 ml-1m 测试误差

Fig. 6 Test error of ml-1 m dataset

实验 2 与基准算法的性能比较

对于 MovieLens 数据集,实验结果均为算法运行 5 次后 5 次结果的平均值;而 Netflix 由于数据量庞大,只通过交叉验证参数后取最优结果。对各算法设置统一的参数进行比较,结果如表 2 所列。

表 2 算法性能的比较

Table 2 Performance comparison with different methods

算法	ml-100 k		ml-1 m		ml-10 m		Netflix	
	RMSE	*MAE*	*RMSE*	*MAE*	*RMSE*	*MAE*	RMSE	MAE
I-RBM-CF	0.938	0.737	0.872	0.766	0.852	0.732	—	—
BiasMF	0.921	0.721	0.856	0.732	0.802	0.682	0.882	0.702
I-AutoRec	0.905	0.712	0.841	0.699	0.791	0.662	0.848	0.672
NNMF	0.923	0.727	0.852	0.712	0.809	0.692	0.852	0.688
DMFN-1	**0.885**	**0.697**	**0.830**	**0.652**	0.806	0.691	0.863	0.692
DMFN-2	0.895	0.701	0.837	0.655	**0.779**	**0.657**	**0.840**	**0.661**

从表 2 可以看出,基于多层网络的算法 AutoRec,NNMF 及 DMFN 的效果比其他浅层方法的效果好;而 DMFN 随着训练数据的增大,其性能变得更加优异。RBM 及 AutoRec 均采用矩阵的行或列作为输入;而本文算法同时将行和列向量作为输入,最后整合其行和列的信息,因此能学习到更多的信息,效果也有所提升。而单层网络在数据集较小(如 ml-100 k)时的效果较好。NNMF 算法直接把潜在特征向量作为未知参数学习,然后交替优化潜在特征向量与网络参数。而本文提出的算法是通过把矩阵的行和列向量做投影,得到潜在特征向量。从实验结果来看,在相等的网络层及参数情况下,DMFN 优于 NNMF。通过投影得到潜在特征向量比直接学习潜在特征向量的效果好,而且能有效减少训练参数,还能提高网络训练的稳定性。而在 ml-10 m 及 Netflix 这两个数据量较大的数据集上,基于深度学习的算法明显优于浅层的分解算法。从实验结果来看,当数据量不大(如 ml-100 k 及 ml-1 m)时,1 层感知器(DMFN-1)能获得比较理想的结果;但当数据增大(如在 ml-10 m 及 Netflix 数据集上)时,DMFN-2 即 2 层感知器能获得较好的结果。

实验 3 潜在特征向量维数对预测结果的影响

该实验主要评估潜在特征向量维数对实验误差的影响。潜在特征向量维数是 DMFN 算法最敏感的参数,不同维数的测试误差如图 7 所示。由图 7 可以看出,测试误差随着潜在特征向量维数的增加而减小;但是维数过高会导致结果不稳定,如对于 ml-100 k 数据集,若取特征维数为 500,则其测试误差如图 8 所示,虽然其最低的测试误差达到 0.877,但结果不稳定,随着训练周期的增大而大幅波动,在训练过程中出现了过拟合现象。因此,在实验中并不是维数越高越好,维数过高同时也会导致训练困难。因此,需要在实验中交叉验证,选取适当的维数。

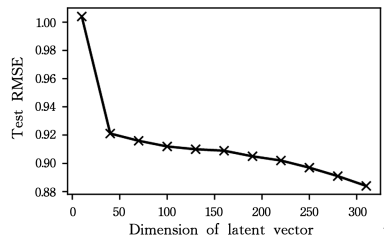

图 7 ml-100k 上潜在特征维数对测试误差的影响

Fig. 7 Test error as dimension of latent vector increases on ml-100 k dataset

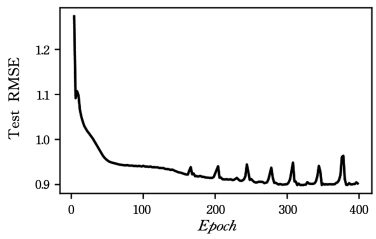

图 8　ml-100k 上潜在向量维数为 500 时的测试误差

Fig. 8　Test errors when dimension of latent vector is set to 500 on ml-100k dataset

实验 4　不同类型的双线性池化性能的比较

3.4 节讨论了 4 种不同类型的双线性池化(pooling)，即式(3)—式(6)，表 3 列出了这几种池化函数对填充结果的影响。该层的主要作用是对网络中的矩阵行和列的输出进行整合，从而得到模型的输出值。通常将双线性映射矩阵用低秩矩阵来表示，这样使得网络更容易训练，且不损失精度。从以上结果可以看出，利用式(4)和式(6)进行池化的效果较好。而层数为 3 时，其填充性能有所下降。对于 ml-1m，用 1 层感知器和式(6)做池化获得的误差最小。同时，各个不同双线性池化方法均比直接使用内积也有一定的性能提升。

表 3　ml-1m 上不同池化的测试结果(*RMSE*)

Table 3　*RMSE* with different pooling operators on ml-1m dataset

池化类型	多层感知器层数			
	0	1	2	3
(3)	0.845	0.836	0.841	0.841
(4)	0.841	0.831	0.838	0.841
(5)	0.842	0.833	0.838	0.841
(6)	0.841	0.831	0.839	0.840
内积	0.846	0.837	0.841	0.845

实验 5　不同激活函数的性能比较

由于本文所提算法的非线性主要来自于对每层施加的激活函数，因此选择适当的激活函数十分必要。本文主要对 Sigmoid，Relu 及 Tanh 进行评估，并考虑同时采用 Relu+Sigmoid(嵌入层用 Relu，多层感知器层用 Sigmoid)的情况，其结果如表 4 所列。由表 4 可知，基于 Sigmoid 的效果远远优于其他激活函数的效果。激活函数通过将线性变换转为非线性变换，提高了网络的非线性能力，选择合适的激活函数对填充效果的影响也十分明显。

表 4　ml-1m 上不同激活函数对结果(*RMSE*)的影响

Table 4　Effect of different activation functions on *RMSE* on ml-1m dataset

激活函数	训练误差	测试误差
Sigmoid	0.811	0.831
Relu	0.872	0.861
Tanh	0.881	0.878
Relu+Sigmoid	0.837	0.841

结束语　本文通过对传统矩阵分解进行扩展，结合深度学习，提出了深度矩阵分解网络。该网络的优点是能够弥补传统矩阵分解模型的不足，提高处理非线性缺损数据的建模能力。首先将输入映射成潜在特征向量，然后结合多层感知器，最后通过双线性池化层对行和列的输出向量进行融合，从而得到模型的预测值。实验表明，本文所提算法的性能不仅比传统的分解方法有较大提升，而且相比基于现阶段的深度学习方法也有所提升。本文只探讨了其在推荐系统中的应用，未来可以将其推广到其他应用领域，如图像填充(Image Inpainting)、运动结构恢复(Structure from Motion)等。

参 考 文 献

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

[2] PAN T T, WEN F, LIU Q R. Collaborative Filtering recommendation algorithm based on rating matrix filling and item predictability[J]. Acta Automatica Sinica, 2017, 43(9): 1597-1606. (in Chinese)
潘涛涛，文锋，刘勤让. 基于矩阵填充和物品可预测性的协同过滤算法[J]. 自动化学报，2017，43(9)：1597-1606.

[3] HU Y, ZHANG D, YE J, et al. Fast and Accurate Matrix completion via truncated nuclear norm regularization [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013, 35(9): 2117-2130.

[4] WANG Y, LEE C M, CHEONG L F, et al. Practical Matrix completion and corruption recovery using proximal alternating robust subspace minimization[J]. International Journal of Computer Vision, 2015, 111(3): 315-344.

[5] CABRAL R S, LA TORRE F D, COSTEIRA J P, et al. Matrix completion for weakly-supervised multi-label Image classification[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015, 37(1): 121-135.

[6] WEN Z, YIN W, ZHANG Y, et al. Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm[J]. Mathematical Programming Computation, 2012, 4(4): 333-361.

[7] JAIN P, NETRAPALLI P, SANGHAVI S, et al. Low-rank matrix completion using alternating minimization[C]// ACM Symposium on Theory of Computing. ACM, 2013: 665-674.

[8] MNIH A, SALAKHUTDINOV R. Probabilistic matrix Factorization[M]// Advances in Neural Information Processing Systems. Berlin: Springer, 2007: 1257-1264.

[9] CANDES E J, RECHT B. Exact matrix completion via convex optimization[J]. Foundations of Computational Mathematics, 2009, 9(6): 717-772.

[10] KRIZHEVSKY A, SUTSKEVER I, HINTON G E, et al. ImageNet classification with deep convolutional neural networks [C]// Advances in Neural Information Processing Systems, 2012, 141(5): 1097-1105.

[11] FU W B, SUN T, LIANG J, et al. Review of principle and application of deep learning [J]. Computer Science. 2018, 45(S1): 11-15. (in Chinese)
付文博，孙涛，梁籍，等. 深度学习原理及应用综述[J]. 计算机科学，2018，45(S1)：11-15.

[12] WANG S, SUN G M, ZOU J Z, et al. Parallel collaborative filtering algorithm based on user recommended Influence[J]. Computer Science, 2017, 44(9): 250-255. (in Chinese)

王硕,孙光明,邹静昭,等.基于用户推荐影响度的并行协同过滤算法[J].计算机科学,2017,44(9):250-255.

[13] SALAKHUTDINOV R,MNIH A,HINTON G. Restricted Boltzmann machines for collaborative filtering[C]// Proceedings of the 24th International Conference on Machine Learning. ACM,2007:791-798.

[14] 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. ACM,2015:111-112.

[15] LI X,SHE J. Collaborative variational autoencoder for recommender systems[C]// Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM,2017:305-314.

[16] WU C Y,AHMED A,BEUTEL A,et al. Recurrent recommender networks[C]// Proceedings of the Tenth ACM International Conference on Web Search and Data Mining. ACM,2017:495-503.

[17] KIM D,PARK C,OH J,et al. Convolutional matrix factorization for document context-aware recommendation[C]// Proceedings of the 10th ACM Conference on Recommender Systems. ACM,2016:233-240.

[18] XUE H J,DAI X,ZHANG J,et al. Deep Matrix Factorization Models for Recommender Systems[C]// Twenty-Sixth International Joint Conference on Artificial Intelligence. AAAI Press,2017:3203-3209.

[19] ZHANG S,YAO L,SUN A,et al. Deep learning based recommender system:A survey and new perspectives[J]. ACM Computing Surveys,2019,52(1):1-38.

[20] CANDÉS E J,TAO T. The power of convex relaxation:Near-optimal matrix completion[J]. IEEE Transactions on Information Theory,2010,56(5):2053-2080.

[21] SI S,CHIANG K Y,HSIEH C J,et al. Goal-directed inductive matrix completion[C]// Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM,2016:1165-1174.

[22] FAN J,CHOW T. Deep learning based matrix completion[J]. Neurocomputing,2017,266(11):540-549.

[23] DZIUGAITE G K,ROY D M. Neural network matrix factorization[J]. arXiv:1511.06443,2015.

[24] FAZEL M. Matrix rank minimization with applications[D]. Palo Alto:Stanford University,2002.

[25] CAI J F,CANDÉS E J,SHEN Z. A singular value thresholding algorithm for matrix completion[J]. SIAM Journal on Optimization,2010,20(4):1956-1982.

[26] LU C,TANG J,YAN S,et al. Generalized nonconvex nonsmooth low-rank minimization[C]// IEEE Conference on Computer Vision and Pattern Recognition. IEEE,2014:4130-4137.

[27] SREBRO N,RENNIE J,JAAKKOLA T S. Maximum-margin matrix factorization[M]// Advances in Neural Information Processing Systems. Berlin:Springer,2005:1329-1336.

[28] LIN Z,XU C,ZHA H. Robust matrix factorization by majorization minimization[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence,2018,40(1):208-220.

[29] BOUMAL N,ABSIL P. RTRMC:A Riemannian trust-region method for low-rank matrix completion[M]// Advances in Neural Information Processing Systems. Berlin:Springer,2011:406-414.

[30] TRIGEORGIS G,BOUSMALIS K,ZAFEIRIOU S,et al. A deep matrix factorization method for learning attribute representations[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence,2017,39(3):417-429.

[31] FAN J,CHENG J. Matrix completion by deep matrix factorization[J]. Neural Networks,2018,98(2):34-41.

[32] HE X,LIAO L,ZHANG H,et al. Neural collaborative filtering[C]// Proceedings of the 26th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee,2017:173-182.