推荐方法之冷启动调研

目录

推荐方法之冷启动

本调研报告针对近两年顶会中发表的推荐算法中的冷启动方法进行一个梳理，主要针对在实际中可能出现的冷启动问题筛选出比较有应用价值的文章进行总结。

1. Low-Rank Linear Cold-Start Recommendation from Social Data

AAAI, 2017

Low-rank Linear Cold-Start Recommendation from Social Data

Suvash Sedhain[†*], Aditya Krishna Menon[*†], Scott Sanner[‡†], Lexing Xie[†*], Darius Braziunas[S]
{[†]Australian National University, [*]Data61}, Canberra, ACT, Australia
[‡] University of Toronto, Toronto, Canada
[S] Rakuten Kobo Inc., Toronto, Canada
suvash.sedhain@anu.edu.au, aditya.menon@data61.csiro.au, ssanner@mie.utoronto.ca, lexing.xie@anu.edu.au, dbraziunas@kobo.com

冷启动问题是指对新用户推荐内容，这些新用户是没有相应的历史偏好信息的。本文中首先说明常用的三种冷启动模型是基于内容的线性模型的特例，随后提出了 LoCo，一种新的冷启动推荐方法包括以下三个方面：a）采用线性回归来学习偏好的权重；b）将权重低秩参数化来克服高维的问题；c）采用随机 SVD 的方法学习低秩权重。

1.1 算法背景

为了解决冷启动问题，需要用到用户的边信息，即属性信息。三种常见的处理个性化推荐的方法如下：

（1）基于内容的过滤（Content-based filtering）：探索边信息或者说辅助信息（side-information）X 和商品偏好 R（用户-商品关系矩阵）的关系；

（2）协同过滤（Collaborative filtering）：探索所有用户间偏好 R 之间的关系，比如通过 k 近邻推荐或者矩阵分解；

对于用户 u 采用 k 近邻方法：$\mathbf{R} \approx \mathbf{S}\mathbf{R}_{\text{tr}}$，其中 S 可以是预先设定的相似度，可以采用余弦相似度来衡量。

（3）混合过滤（Hybrid filtering）：探索两种形式的相关性。

可以将用户分为暖启动用户 U_{tr}（至少有一次购买行为）和冷启动用户 U_{te}，同时有购买矩阵 R_{tr} 和 R_{te}，其中 $R_{te} = 0$。我们希望预测出$\hat{R}_{te}$。但是我们有辅助信息矩阵 X_{tr} 和 X_{te}，因此我们希望分析训练数据中 X_{tr} 和 R_{tr} 的关系，来根据 X_{te} 预测出$\hat{R}_{te}$。

现有的推荐算法在冷启动方面的应用都是根据辅助矩阵 X_{te} 采用线性模型进行预测：

$$\hat{\mathbf{R}}_{\text{te}} = \mathbf{X}_{\text{te}}\mathbf{W}$$

1.1.1 Social neighbourhood model：Neighbourhood + metadata similarity

$$\hat{\mathbf{R}}_{\text{te}} = \mathbf{X}_{\text{te}} \odot \mathbf{X}_{\text{tr}}^{T} \star \mathbf{R}_{\text{tr}}$$

其中，$\mathbf{S} = \mathbf{X}_{\text{te}} \odot \mathbf{X}_{\text{tr}}^{T}$，认为辅助信息之间的相似性和购买矩阵间的相似性

相同。

$\hat{\mathbf{R}}_{\text{te}} = \mathbf{X}_{\text{te}}(\mathbf{X}_{\text{tr}}^T\mathbf{R}_{\text{tr}})$，因此：$\mathbf{W} = \mathbf{X}^T\mathbf{R}$

1.1.2 CMF model：Matrix factorisation with regularisation

$$\min_{\mathbf{U},\mathbf{V},\mathbf{Z}} ||\mathbf{R} - \mathbf{U}\mathbf{V}||_F^2 + \mu||\mathbf{X} - \mathbf{U}\mathbf{Z}||_F^2 + \Omega(\mathbf{U},\mathbf{V},\mathbf{Z}),$$

$$\Omega(\mathbf{U},\mathbf{V},\mathbf{Z}) = \frac{\lambda_U}{2}||\mathbf{U}||_F^2 + \frac{\lambda_V}{2}||\mathbf{V}||_F^2 + \frac{\lambda_Z}{2}||\mathbf{Z}||_F^2.$$

采用了一个公用的潜在矩阵 U 作为用户-商品关系矩阵 R 和辅助信息 X 矩阵分解时候的公用项。

$$\hat{\mathbf{R}}_{\text{te}} = \mathbf{U}_{\text{te}}\mathbf{V}$$

V 根据训练数据可以获得，U_{te} 要根据测试数据 X_{te} 计算获得。

可以根据上面的目标函数得到$\mathbf{U} = (\mathbf{R}\mathbf{V}_*^T + \mathbf{X}\mathbf{Z}_*^T)(\mathbf{V}_*\mathbf{V}_*^T + \mathbf{Z}_*\mathbf{Z}_*^T)^{-1}$，其中 U，V 和 Z 是迭代计算得到的，假设得到了最优的 V 和 Z，可以得到 U 的解析表达。对于测试数据，根据 X_{te} 计算 U_{te}，其中 U 是根据测试数据改变，而 V 和 Z 是训练数据得到的

$$\hat{\mathbf{R}}_{\text{te}} = \mathbf{U}_{\text{te}}\mathbf{V}_* = \mathbf{X}_{\text{te}}\mathbf{Z}_*^T(\mathbf{V}_*\mathbf{V}_*^T + \mathbf{Z}_*\mathbf{Z}_*^T)^{-1}\mathbf{V}_*$$

因此：$\mathbf{W} = \mathbf{Z}_*^T(\mathbf{V}_*\mathbf{V}_*^T + \mathbf{Z}_*\mathbf{Z}_*^T)^{-1}\mathbf{V}_*$

1.1.3 BPR-LinMap model：Matrix factorisation with feature mapping

首先采用矩阵分解的思想，$\mathbf{R}_{\text{tr}} \approx \hat{\mathbf{R}}_{\text{tr}} = \mathbf{U}_{\text{tr}}\mathbf{V}$，随后计算辅助矩阵 X_{tr} 和矩阵 U_{tr} 之间的关系，如下式所示：

$$\min_{\mathbf{T}} \|\mathbf{U}_{\text{tr}} - \mathbf{X}_{\text{tr}}\mathbf{T}\|_F^2 + \frac{\lambda_T}{2}\|\mathbf{T}\|_F^2$$

根据这个关系，可以在测试数据时根据辅助矩阵 X_{te} 得到矩阵 U_{te}，$U_{te} = X_{te}T$ 因此：

$$\hat{\mathbf{R}}_{\text{te}} = \mathbf{U}_{\text{te}}\mathbf{V}_* = \mathbf{X}_{\text{te}}\mathbf{T}_*\mathbf{V}_*$$

$$\mathbf{T}_* = (\mathbf{X}_{\text{tr}}^T\mathbf{X}_{\text{tr}} + \lambda_T\mathbf{I})^{-1}\mathbf{X}_{\text{tr}}^T\mathbf{U}_{\text{tr}}$$

因此：$\mathbf{W} = (\mathbf{X}_{\text{tr}}^T\mathbf{X}_{\text{tr}} + \lambda_T\mathbf{I})^{-1}\mathbf{X}_{\text{tr}}^T\hat{\mathbf{R}}_{\text{tr}}$

1.1.4 优缺点分析

Social neighbourhood 模型方便计算，但是 W 没有通过明确的目标函数得到，同时没有解释特征之间的相关性，有欠拟合的风险；

BPR-LinMap 解释了特征间的相关性，但是可能过拟合。同时当 X 维数很高时，求解 T 时会比较耗时，而网络当中的维数通常是非常高的；

CMF 隐式地进行了元数据矩阵的低秩分解，因此可以适用于高维数据，但

是它由很多超参数组成，参数的调整影响了最终的性能。

1.2 算法原理：LoCo

根据上面的分析，本文提出的方法就是直接构建商品偏好 R（用户-商品关系矩阵）和辅助信息 X 的关系，首先对训练数据建立目标函数：

$$\min_{\mathbf{W}} ||\mathbf{R}_{\text{tr}} - \mathbf{X}_{\text{tr}}\mathbf{W}||_F^2 + \frac{\lambda}{2}||\mathbf{W}||_F^2$$

则：

$$\mathbf{W} = (\mathbf{X}_{\text{tr}}^T\mathbf{X}_{\text{tr}} + \lambda\mathbf{I})^{-1}\mathbf{X}_{\text{tr}}^T\mathbf{R}_{\text{tr}}$$

那么：

$$\hat{\mathbf{R}}_{\text{te}} = \mathbf{X}_{\text{te}}\mathbf{W}$$

当然，上述的解法存在一定的问题，因此对上述的方法进行了改进：

(1). 引入了低秩的约束：

$$\min_{\mathbf{W}:\ \text{rank}(\mathbf{W})\le K} ||\mathbf{R}_{\text{tr}} - \mathbf{X}_{\text{tr}}\mathbf{W}||_F^2 + \frac{\lambda}{2}||\mathbf{W}||_F^2$$

其中 K<< min (U, I)。引入低秩可以避免虚假的关系预测。

(2). 进行低秩矩阵参数的子集化

上述的目标函数是非凸的，为了克服这个问题，提出了低秩矩阵参数的子集化的方法，对 X_{tr} 进行 SVD 分解：$\mathbf{X}_{\text{tr}} \approx \mathbf{U}_K\mathbf{S}_K\mathbf{V}_K^T$，这不是完全的 SVD 分解，只是选取了秩为 K 的 SVD 分解。此时假设 $W = V_K Z$，那么很明显 $\text{rank}(W) \le K$。此时的优化目标为：

$$\min_{\mathbf{Z}} ||\mathbf{R}_{\text{tr}} - \mathbf{X}_{\text{tr}}\mathbf{V}_K\mathbf{Z}||_F^2 + \frac{\lambda}{2}||\mathbf{Z}||_F^2$$

其中，V_K 也是已知的，只有 Z 是待求的参数。

$$\mathbf{Z} = (\mathbf{V}_K^T\mathbf{X}_{\text{tr}}^T\mathbf{X}_{\text{tr}}\mathbf{V}_K + \lambda\mathbf{I})^{-1}\mathbf{V}_K^T\mathbf{X}_{\text{tr}}^T\mathbf{R}_{\text{tr}}$$

因此，

$$\mathbf{W} = \mathbf{V}_K(\mathbf{V}_K^T\mathbf{X}_{\text{tr}}^T\mathbf{X}_{\text{tr}}\mathbf{V}_K + \lambda\mathbf{I})^{-1}\mathbf{V}_K^T\mathbf{X}_{\text{tr}}^T\mathbf{R}_{\text{tr}}$$

(3). 最后采用随机 SVD 的方法进行 SVD 的近似计算。

下表是对这些方法中 W 的总结：

Method	Weight W
Social neighbourhood	$\mathbf{X}_{\text{tr}}^T\mathbf{R}_{\text{tr}}$
CMF	$\mathbf{Z}_*^T(\mathbf{V}_*\mathbf{V}_*^T + \mathbf{Z}_*\mathbf{Z}_*^T)^{-1}\mathbf{V}_*$
BPR-LinMap	$(\mathbf{X}_{\text{tr}}^T\mathbf{X}_{\text{tr}})^{-1}\mathbf{X}_{\text{tr}}^T\hat{\mathbf{R}}_{\text{tr}}$
Linear regression	$(\mathbf{X}_{\text{tr}}^T\mathbf{X}_{\text{tr}})^{-1}\mathbf{X}_{\text{tr}}^T\mathbf{R}_{\text{tr}}$
LoCo	$\mathbf{V}_K(\mathbf{V}_K^T\mathbf{X}_{\text{tr}}^T\mathbf{X}_{\text{tr}}\mathbf{V}_K)^{-1}\mathbf{V}_K^T\mathbf{X}_{\text{tr}}^T\mathbf{R}_{\text{tr}}$

总结一下所提方法的创新点：

1）提出了一种低秩参数化的线性矩阵权重；
2）采用随机 SVD 的方法对 X_{tr} 进行分解；
3）采用多元线性回归的方法计算权重

2. From Zero-Shot Learning to Cold-Start Recommendation

AAAI, 2019

From Zero-Shot Learning to Cold-Start Recommendation

Jingjing Li[1], Mengmeng Jing[1], Ke Lu[1], Lei Zhu[2], Yang Yang[1], Zi Huang[3]
[1] University of Electronic Science and Technology of China
[2] Shandong Normal University; [3]The University of Queensland
lijin117@yeah.net; jingmeng1992@gmail.com; kel@uestc.edu.cn; leizhu0608@gmail.com

零样本学习与推荐冷启动是计算机视觉和推荐系统中的两大难题。这两个问题在不同的研究领域中，所以一般是分开研究的。在这篇论文中，把零样本学习与冷启动问题作为同一个问题的两个扩展问题来看待。比如，他们都在试图对未知的分类进行预测，一个用于特征表示，一个用于描述的补充。然而目前还没有从零样本学习的角度去解决冷启动推荐的问题。本论文是首次将推荐冷启动问题看做是零样本学习的问题，并提出了一个自定义的零样本学习方法去处理冷启动问题。作者提出了一个**低秩线性自编码器**(LLAE)，为了解决以下三个问题：域迁移、虚假相关和计算效率。在这篇论文中，**LLAE 由两部分组成：一个是用低秩编码器将用户行为空间映射到用户属性空间；另一个是用解码器从用户属性中重构出用户行为**。在零样本学习与冷启动的大量实验中验证了该方法是一个双赢的方法。与传统几种方法对比，零样本学习能够显著提高冷启动的性能。

这里简单阐述下**零样本学习**（ZSL, Zero-shot learning）的基本思想：

在传统的分类模型中，为了解决多分类问题（例如三个类别：猫、狗和猪），就需要提供大量的猫、狗和猪的图片用来模型训练，然后给定一张新的图片，就能判定属于猫、狗或猪的其中哪一类。但是对于之前训练图片未出现的类别（例如牛），这个模型便无法将牛识别出来，而 ZSL 就是为了解决这种问题，即在训练中未出现的类别如何在训练当中判断出来。这里的判断并不是单纯的指出测试样本和原有的训练样本是不同的，而是直接的明确说出这是哪种动物，比如如果在测试样本中有熊猫这类动物，那么测试的时候就要明确说出这是熊猫，而不是说出现了一个新的类别。

这个问题看上去好像不可能，因为训练中没有提供新类别的训练数据，在测试中如何将其识别出来呢？这里就用到了各个类别的属性特征。举个例子，我们现在要对动物进行分类，动物的属性特征就包括：是否有尾巴、有几只脚、颜色、是否有角等等属性特征。那么对于每一类的动物，我们都可以对这类动物的属性特征进行统计，得到特定动物的属性特征的集合。在进行训练的时候就是将动物的图像映射到对应的属性特征上，然后根据得到的属性特征再判断动物最后的类别。

这里以零样本学习中常见的**直接属性预测（DAP , Direct attribute prediction）**进行说明。

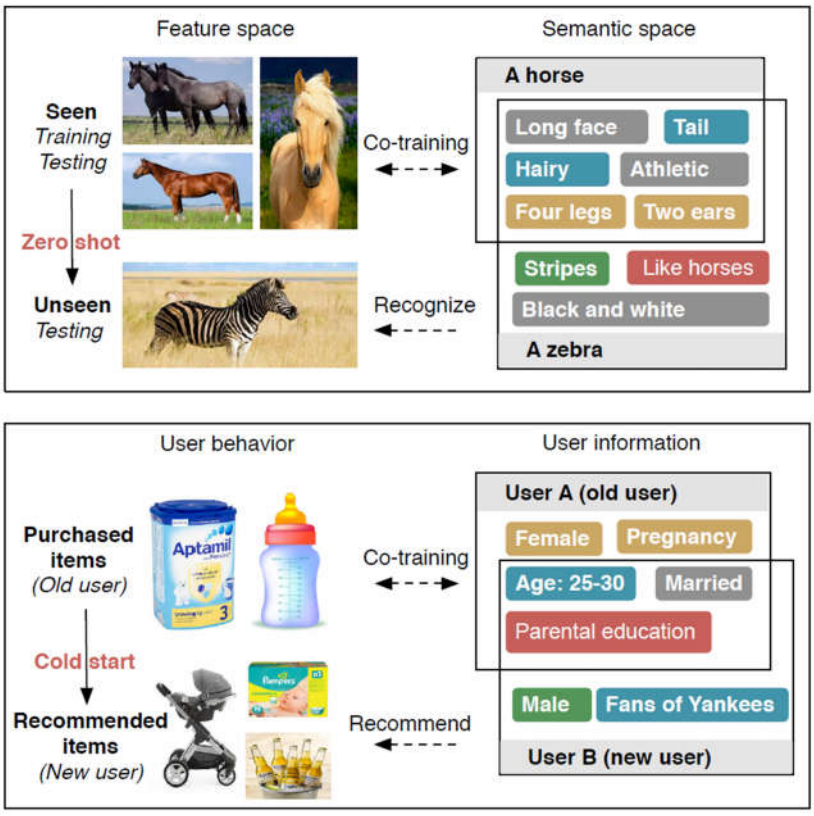
DAP 可以理解为一个三层模型：第一层是原始输入层，例如一张电子图片（可以用像素的方式进行描述）；第二层是 p 维特征空间，每一维代表一个属性

特征（例如是否有尾巴、是否有毛等等）；第三层是输出层，输出模型对输入样本的类别进行判断。在第一层和第二层中间，**训练 p 个分类器**，用于对一张图片判断是否符合 p 维特征空间各个维度所对应的特征，即判断是否有尾巴，有没有脚等等属性特征；最后在第二层和第三层间，有一个语料知识库，用于保存 p 维特征空间和输出 y 的对应关系，这个语料知识库是事先人为设定的。

假设我们已经训练好了一个 DAP 模型，第一层和第二层间的分类器可以判断出是否黑眼圈、是否喜欢吃竹子之类的特征，然后在语料知识库里面包含一个映射：黑眼圈、喜欢吃竹子--> 熊猫，那么即使我们的模型在训练时没有见过熊猫的图片，在遇到熊猫的图片时，我们可以直接通过图片得到这个图片的属性信息，即有黑眼圈、喜欢吃竹子，然后结合知识语料库中包含这些属性特征的类别，来判断出这张图片是熊猫。我们也可以通过计算熊猫图片的属性特征与其他训练样本的属性特征的距离，得到熊猫和哪种动物比较相似的信息。整个 DAP 的运作思想就是类似于上述过程。

因此零样本学习就是通过学习输入数据的属性特征，然后得到数据与属性特征间的映射关系，在测试数据中，我们根据已有数据可以得到测试数据的属性特征，最后根据属性特征进行分类。这里需要事先知道的就是属性特征和类别之间的关系，同时也需要训练数据尽可能的包含到测试数据的属性特征。例如，如果训练数据不包含一些测试数据分类必要的属性特征，例如如果训练数据都是陆地动物猫狗猪之类的，不包含是否生活在水中这一属性特征，那么就很难分辨出鱼这种动物。

2.1 算法背景



零样本学习和冷启动推荐示例

推荐系统基于用户当前以及过去的行为信息来给用户推荐可能喜欢的商品。如果不能获取一个用户过去的行为信息，或者他就是一个新用户。这时对于大多数的推荐系统，特别是那些基于协调过滤的主流推荐系统就无法使用了。所以大家提出了各种不同的方法来解决这个问题，这就是众所周知的冷启动推荐（CSR）问题。最近像类似的跨域信息、个人信息以及社交网络数据会被用来解决 CSR 问题。

现有冷启动方法，是利用用户偏好为新用户生成推荐。在冷启动推荐中，使用的用户偏好有两个维度，一个是用户属性（用户偏好与用户信息），一个是用户行为（购买行为与过去的互动信息）。属性维度用于描述用户的兴趣，行为维度用于表示用户在目标系统中的交互。**因此冷启动推荐可以定义为，在没有用户行为信息，而只有用户的部分属性信息的情况下，对新用户进行推荐的问题。**我们假设有相同兴趣（属性维度）的人，在消费上的行为也是相似的，那么冷启动问题就可以按照下面两步处理：

1）把用户行为属性映射到属性空间中，这样就能把新用户与老用户关联起来。

2）根据用户属性重构用户行为，这样我们就可以给新用户进行推荐了。

从上图中可以看出，零样本与冷启动推荐是一个相同概念的两个扩展，它们都包含两个空间：一个是特征的表示，一个是描述的补充。它们都试图利用可见与不可见的描述空间来预测特征空间中不可见的情况。（文章中的特征空间是指的输入数据的图片或者用户的行为空间，描述空间：零样本指的是语义空间，冷启动指的是用户属性信息）。本文中首次提出使用零样本学习方法解决冷启动问题。

将冷启动定义为零样本学习问题，在本文中我们挑战了三个难题：

1）领域转移问题：不仅用户的行为空间与属性空间是异构的，而且老用户与新用户的概率分布也是发散的。因此，我们必须保证用户行为是可以通过用户属性进行重构的。**解决方法：encoder-decoder**

2）冷启动推荐中的用户行为与计算机视觉任务中的零样本学习不同之处在于，它非常的稀疏。在现实零售巨头比如亚马逊，他有上亿的用户与更加丰富的商品，对于一个特定的用户，在系统中只有很少的交互行为，与商品项之间的交互可能会更少。因此用户-商品的矩阵会非常的稀疏。**解决方法：低秩约束**

3）效率问题，推荐系统是一个在线系统，用户讨厌等待。**解决方法：线性模型**

因此，本文提出了一种新的零样本学习（ZSL）来解决冷启动（CRS）问题，名字叫低秩线性自动编码器（LLAE）。这个是基于 encoder-decoder 模式。**LLAE 由编码器和解码器组成，编码器将用户行为空间映射到了用户属性空间；解码器通过用户的属性空间重构用户行为。**重构部分保证可以从用户属性生成用户行为。由于效率问题我们把 LLAE 表示成线性模型。模型计算量与样本数无关，因此它可以用来处理大规模数据集。并采用低秩约束来处理稀疏问题。低秩表示已经被证明可以有效的从损坏的观测数据中揭示真实数据。一个行为可能与多个属性相关联，关联的这些属性也有不同的权重，有些属性是不重要的。如果我们把所有属性都考虑进去了，可能会削弱主因素带来的影响，引入过度拟合，降低泛化能力。低秩约束由于其数学上的特性，能够找出主导因素，过滤掉噪点联系，换句话说，就是防止了那些虚假关联因素。低秩约束还可以从域适应的角度上来对齐域转移。因此，本文的贡献点如下：

1）首次将零样本学习与冷启动关联起来；
2）提出了 LLAE 模型，效率高，能处理大规模数据。
3）实验结果表明了这个方法的有效性

2.2 算法原理

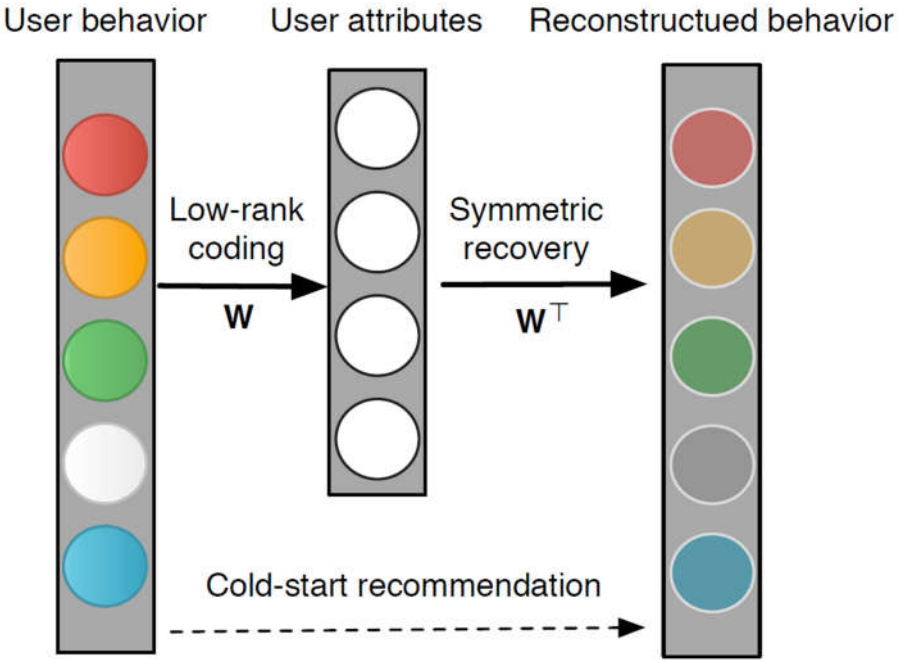
2.2.1 Linear Low-rank Denoising Autoencoder

给定一个输入数据矩阵 X（用户行为），假设我们可以学习到一个映射 W，它能将矩阵 X 投影到一个潜在的空间 S（用户属性）中。另外一个映射 M，它能够从用户属性空间 S 中重构出用户行为 X。作为一个优化问题，我们的目标就是最小化重构误差。因此目标函数如下：

$$\min_{\mathbf{W},\mathbf{M}} \|\mathbf{X}-\mathbf{MWX}\|_F^2, \ \ s.t.\ \mathbf{WX}=\mathbf{S}$$

通常情况下，潜在空间都是用隐藏层来表示，考虑效率与可解释性，这里只在模型中部署一个隐藏层 S。在这个文章中，S 有明确的定义，在零样本学习 ZSL 中表示语义空间，在冷启动推荐 CSR 中表示用户侧信息。捆绑权值（tied weights）被引入到自编码器中，可以加快模型训练，减少模型参数。文章中，考虑捆绑权重 $M=W^T$，由此可以得到下式：

$$\min_{\mathbf{W}} \|\mathbf{X}-\mathbf{W}^\top\mathbf{WX}\|_F^2, \ \ s.t.\ \mathbf{WX}=\mathbf{S}$$



LLAE 示意图

上图是 LLAE 一个简单说明，首先我们学习一个低秩编码器，我们把用户行为映射到用户属性空间。然后利用新用户的属性重构用户行为。为了提高效率，编码器与解码器的参数权重是对称的，这里就是利用了 tied weights 算法，注意：编码器可以保证热用户和冷用户在属性空间中能够比较，重构阶段是基于用户属性重构用户行为。

在真实推荐系统中我们面临的挑战是要处理高维的稀疏矩阵。因为有数百万商品、用户，对于一个用户只会与很少的商品发生交互。为了避免由于映射矩阵

带来的伪相关，W 矩阵应该是一个低秩的，所以我们得到下面公式：

$$\min_{\mathbf{W}} \|\mathbf{X} - \mathbf{W}^\top \mathbf{W}\mathbf{X}\|_F^2 + \beta \text{rank}(\mathbf{W}), \ \ s.t.\ \mathbf{W}\mathbf{X} = \mathbf{S}$$

其中 rank(•)是矩阵秩的运算操作，其中，$\beta > 0$ 是一个惩罚参数，值得注意的是，W 上的秩约束至少从两个方面来说对模型都是有益的。

1）过滤从行为空间到属性空间映射的虚假关联；

2）它能够凸显不同用户之间共享属性。

对于一个特定属性的例子，比如：篮球迷，这个属性是不同年龄用户共有的属性。那么在 W 上的低秩约束就能够识别出这些常见的属性。

在一些冷启动任务中，用户行为空间和属性空间这两个空间可能不怎么相关，矩阵低秩约束能够帮助找出相对来说更加相关的部分。在重构阶段更加关键，因为如果没有重构约束的话，映射可能是不准确的。重构部分有效的解决了域转移的问题。尽管用户的行为可能会从温用户变成冷用户，但是从属性到行为的更真实的重构中是能够在温域和冷域中概括出来的。从而使学习到的映射不容易受到领域迁移的影响。

对于 W 的低秩约束使得优化变得困难，因为低秩是众所周知的 NP 难度问题。作为一个替代的方法，采用一个显示形式的低秩约束：

$$\begin{aligned} &\min_{\mathbf{W}} \|\mathbf{X} - \mathbf{W}^\top \mathbf{W}\mathbf{X}\|_F^2 + \beta \sum_{i=r+1}^{d} (\sigma_i(\mathbf{W}))^2 \\ &s.t.\ \mathbf{W}\mathbf{X} = \mathbf{S}, \end{aligned}$$

其中σ_i(W)是 W 的第 i 个奇异值。d 是 W 所有奇异值的数量。不同于迹范数，$\sum_{i=r+1}^{d}(\sigma_i(\mathbf{W}))^2$明确解决了映射 W 矩阵的最小的 r 个奇异值的平方和最小化的问题。

由于：

$$\sum_{i=r+1}^{d}(\sigma_i(\mathbf{W}))^2 = \text{tr}(\mathbf{V}^\top \mathbf{W}\mathbf{W}^\top \mathbf{V})$$

其中 tr(•)是矩阵迹的计算方式。**V 是由 $\mathbf{W}\mathbf{W}^\mathbf{T}$ 矩阵的（d-r）个最小奇异值对应的奇异向量组成。**

目标函数最终可以转变为：

$$\begin{aligned} &\min_{\mathbf{W},\mathbf{V}} \|\mathbf{X} - \mathbf{W}^\top \mathbf{W}\mathbf{X}\|_F^2 + \beta \text{tr}(\mathbf{V}^\top \mathbf{W}\mathbf{W}^\top \mathbf{V}) \\ &s.t.\ \mathbf{W}\mathbf{X} = \mathbf{S}. \end{aligned}$$

最后，为了学习到更加稳定的隐藏层，这里通过引入噪点数据的输入训练了一个去噪的自动编码器。这里把输入的 X 中 10%的值设置为 0。得到一个带有噪声的样本输入$\hat{\mathbf{X}}$。这个时候优化的最终目标如下：

$$\begin{aligned} &\min_{\mathbf{W},\mathbf{V}} \|\mathbf{X} - \mathbf{W}^\top \mathbf{W}\mathbf{X}\|_F^2 + \beta \text{tr}(\mathbf{V}^\top \mathbf{W}\mathbf{W}^\top \mathbf{V}) \\ &s.t.\ \mathbf{W}\hat{\mathbf{X}} = \mathbf{S}. \end{aligned}$$

注意，带噪音的$\hat{\mathbf{X}}$是在约束项中的。

2.2.2 模型优化

为了优化上式，首先改写为：

$$\min_{\mathbf{W},\mathbf{V}} \|\mathbf{X}-\mathbf{W}^\top\mathbf{S}\|_F^2+\beta\mathrm{tr}(\mathbf{V}^\top\mathbf{W}\mathbf{W}^\top\mathbf{V}),\ s.t.\ \mathbf{W}\widehat{\mathbf{X}}=\mathbf{S}$$

就是将 S 带入到之前的公式中，S 就是编码之后的用户属性信息。但是上式依然难以优化。这里放宽约束，可以得到如下公式：

$$\min_{\mathbf{W},\mathbf{V}} \|\mathbf{X}-\mathbf{W}^\top\mathbf{S}\|_F^2+\lambda\|\mathbf{W}\widehat{\mathbf{X}}-\mathbf{S}\|_F^2+\beta\mathrm{tr}(\mathbf{V}^\top\mathbf{W}\mathbf{W}^\top\mathbf{V})$$

对 W 求偏导并使其为 0：

$$\begin{aligned}&-\mathbf{S}(\mathbf{X}^\top-\mathbf{S}^\top\mathbf{W})+\lambda(\mathbf{W}\mathbf{X}-\mathbf{S})\widehat{\mathbf{X}}^\top+\beta\mathbf{V}\mathbf{V}^\top\mathbf{W}=\mathbf{0},\\ &\Rightarrow\ (\mathbf{S}\mathbf{S}^\top+\beta\mathbf{V}\mathbf{V}^\top)\mathbf{W}+\lambda\mathbf{W}\mathbf{X}\widehat{\mathbf{X}}^\top=\mathbf{S}(\mathbf{X}^\top+\lambda\widehat{\mathbf{X}}^\top)\end{aligned}\quad (11)$$

值得注意的是，W 的优化涉及到 V，这里优化选择交替更新方式，我们首先将 V 看成常数，根据上式计算 W。根据$\sum_{i=r+1}^{d}(\sigma_i(\mathbf{W}))^2=\mathrm{tr}(\mathbf{V}^\top\mathbf{W}\mathbf{W}^\top\mathbf{V})$更新 V。最后做个简单的替换：

$$\begin{cases}\mathbf{A}=\mathbf{S}\mathbf{S}^\top+\beta\mathbf{V}\mathbf{V}^\top\\ \mathbf{B}=\lambda\mathbf{X}\widehat{\mathbf{X}}^\top\\ \mathbf{C}=\mathbf{S}\mathbf{X}^\top+\lambda\mathbf{S}\widehat{\mathbf{X}}^\top\end{cases}$$

式（11）改写为：

$$\mathbf{A}\mathbf{W}+\mathbf{W}\mathbf{B}=\mathbf{C}\quad (13)$$

下图是算法的伪代码：

Algorithm 1. *Low-rank Linear AutoEncoder for CSR*

Input: User behavior space $\mathbf{X}$, user attribute space $\mathbf{S}$, parameters λ and β.

Output: Recommended items for new users.

Warm-up:

Repeat

1. Solve the eigen-decomposition problem to get $\mathbf{V}$:

$$\mathbf{V}\leftarrow\mathrm{svd}(\mathbf{W}\mathbf{W}^\top).$$

2. Optimize the encoder $\mathbf{W}$ (and the decoder $\mathbf{W}^\top$):

$$\mathbf{W}=\mathrm{sylvester}(\mathbf{A},\mathbf{B},\mathbf{C}),$$

where $\mathbf{A}=\mathbf{S}\mathbf{S}^\top+\beta\mathbf{V}\mathbf{V}^\top,\ \mathbf{B}=\lambda\mathbf{X}\widehat{\mathbf{X}}^\top,\ \mathbf{C}=\mathbf{S}\mathbf{X}^\top+\mathbf{S}\widehat{\mathbf{X}}^\top.$

Until Convergence

Cold-start:

$\mathbf{X}_{new}=\mathbf{W}^\top\mathbf{S}_{new}.$

Recommendation:

Using the logistic regression function to predict the recommendation probability of items, and recommend the top-k items.

2.2.3 算法复杂度

该算法的计算量有两部分组成：

1）W 的优化 2）更新 V，这两个操作的复杂度都是 O(d^3)。然而，如果直

接计算 VV^T 而不是 V，2）的复杂度可以降为 $O(r^2d)$（r<<d 是 W 的秩）。在任何情况下，这个算法只与向量维度相关，与样本数量无关，所以可以用户大规模计算。

2.2.4 零样本分类

给定训练数据 X 与语义表示的 S，我们可以根据公式 13 学习到一个编码的 W 与一个解码的 W^T。对于新的测试样本集 X_{new}，我们可以用 W 将它嵌入到语义空间 $S_{new} = WX_{new}$。这样 X_{new} 的标签就可以通过计算 S_{new} 与 S_{proto} 间的距离得到。S_{proto} 是语义空间中的原型，即各个语义描述的类别（其实你简单的理解就是将 X 转变为语义空间中的 S，看在 S 空间中对应属于哪一类，在 S 空间中有对应的对每一类详细的描述，这是先验知识得到的）。

$$f(\mathbf{X}_{new}) = \arg\min dis(\mathbf{S}_{new}, \mathbf{S}_{proto})$$

$f(\mathbf{X}_{new})$是分类器，它可以返回 X_{new} 的标签，dis 是一个度量距离的方法

2.2.5 冷启动推荐

假设我们用 X 表示用户的行为，即：用户浏览、购买、分享行为，这个通常是一个用户-商品矩阵。S 是用户属性，例如用户的偏好、个人信息和社交网络数据。我们可以根据公式 13 学习到一个编码的 W 与一个解码的 W^T。对于一个新用户，冷启动推荐的目标是学习到一个 X_{new}，**这个 $\mathbf{X}_{new}$ 是用户与商品之间的一个潜在关系，即预测的是否会浏览、购买、分享等行为**。可以通过下面计算得到：

$$\mathbf{X}_{new} = \mathbf{W}^\top \mathbf{S}_{new}$$

2.3 算法应用

2.3.1 零样本学习

结果如下：

Table 3: Accuracy (%) of zero-shot recognition. The best results are marked as bold numbers.

Method	aP&aY	AwA	CUB	SUN	Avg.
DAP	38.23	60.51	39.14	71.92	52.45
ESZSL	24.37	75.31	48.75	82.12	57.64
SSE	46.22	76.35	30.49	82.51	58.89
JLSE	50.46	80.51	42.83	83.86	64.42
LESD	**58.83**	76.62	56.25	88.36	70.02
Ours	56.16	**85.24**	**61.93**	**92.07**	**73.85**

2.3.2 冷启动推荐

结果如下：

Table 4: CSR results of mAP@100 (%) on different datasets.

Method	Flickr	BlogCatalog	YouTube	LastFM
CBF-KNN	28.05	32.71	34.21	17.12
Cos-Cos	31.42	41.06	46.67	12.26
BPR-Map	21.59	28.22	30.35	8.85
CMF	21.41	27.76	28.16	8.13
LoCo	33.57	45.35	48.79	18.09
Ours	**39.25**	**50.13**	**52.45**	**23.07**

2.4 思考

如何将零样本学习的思路用到图推荐中呢？由于新用户和之前的用户或者商品没有交互，那么可用的信息仍然是用户的属性信息，即输入时用户的属性信息的表征。这里，可以考虑对历史用户的 embedding 和属性之间做投影，而用户和商品的交互评分由用户的 embedding 和商品的 embedding 得到，那么这种思路就是将新用户的属性特征->新用户的 embedding->用户和商品的交互评分。

其次，再扩展一下，如果随着用户的行为数据越来越多，如何来更新这个现有的模型呢？重新建模的代价必然很大，那么采用增量学习？这个问题其实是动态图的一个微观实例，既然有 RNN，那么可不可以将其扩展到 GCN，得到 RGCN 呢？（Recurrent Graph Convolutional Network？）

3. Sequential Scenario-Specific Meta Learner for Online Recommendation

KDD, 2019

Sequential Scenario-Specific Meta Learner for Online Recommendation

Zhengxiao Du[1], Xiaowei Wang[2], Hongxia Yang[2], Jingren Zhou[2], Jie Tang[1]
[1] Department of Computer Science and Technology, Tsinghua University
[2] DAMO Academy, Alibaba Group
duzx16@mails.tsinghua.edu.cn,{daemon.wxw,yang.yhx,jingren.zhou}@alibaba-inc.com,jietang@tsinghua.edu.cn

3.1 算法背景

对于一般的推荐场景来说，冷启动问题是个长期存在的挑战。大多数推荐算法依赖于大量的观察数据，对于没什么联系的数据，这些推荐算法产生不了足够好的效果。为了解决这个问题，文章使用了 few-shot learning 以及 meta learning，提出了 Scenario-specific Sequential Meta learner。元学习器可以产生一个一般的初始化模型，这个模型是通过整合许多预测任务信息，同时可以有效的根据具体任务来更新参数。

在淘宝中，大多数促销活动在推出几天甚至几小时内就结束了，因此没有足够的时间来收集用户的信息来进行训练。因此，在有限的观测值中训练特定场景的推荐器是非常重要的。同时，对于一个新的场景而言，超参数对最后的性能有很大的影响，而对于不同的场景而言，超参数又会存在很大的不同，因此找到一个合适的超参数组合通常也需要很大的人力。

因此，本文针对冷启动问题，结合小样本学习（few-shot learning）和元学习（meta-learning），提出了在有限的训练数据下建立元学习器来获取具有良好泛化

能力的推荐器，名为特定场景下的序列元学习器（Sequential Scenario-Specific Meta Learner, s^2 Meta）。它主要包括三个步骤：

1）元学习器自动初始化推荐器以适用于多种场景；

2）利用灵活的更新策略来微调参数；

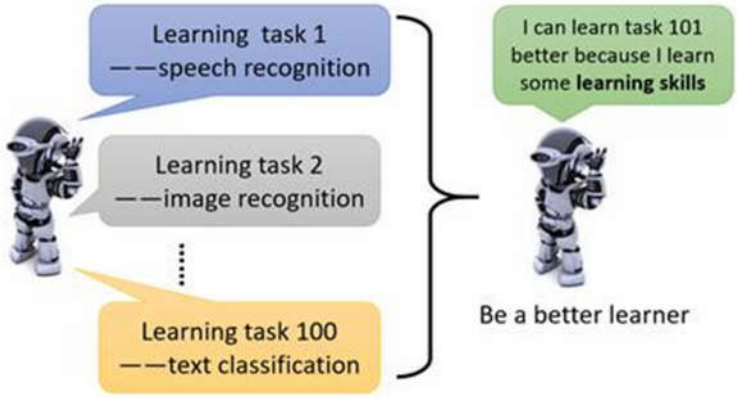
3）及时停止学习以避免过拟合。

3.2 Meta Learning

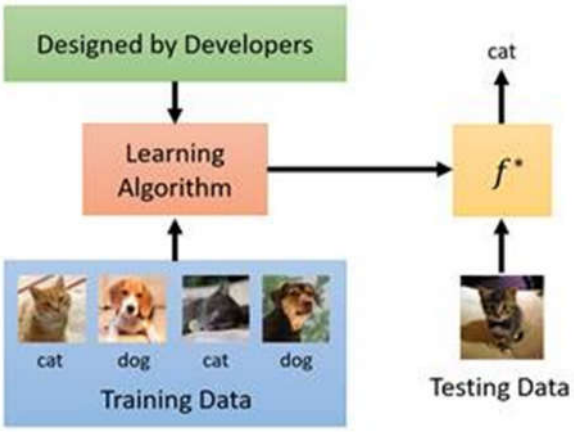
Meta Learning 被称作元学习，不同于 Machine Learning 的目标是让机器能够学习，Meta Learning 则是要让机器学会如何去学习。

举例来说，机器已经在过去的 100 个任务上进行了学习，现在我们希望，机器能够基于过去 100 个任务学习的经验，变成一个更厉害的学习者，这样当在第 101 个新任务到来之时，机器能够更快地学习。值得注意的是，机器之所以能够学习地更快并不是依赖于在旧任务中已获取的“知识”，而是机器学到了如何去更好获取知识的方法，并将这一方法应用于新任务当中，从而较快地提升学习效率。

以上图为例，假设前 99 个学习任务都是各种辨识任务，例如语音辨识、图像辨识等，在前 99 个任务学习完成之后，我们给机器一个新的学习任务，而这个新的学习任务与前 99 个任务没有任何关联，譬如是一个文本分类任务。而现在，Meta Learning 的目的就是希望能够通过前 99 个辨识任务的学习让机器在新的文本分类任务上学习得更好，也就是说，机器在前面的学习中不仅仅学到了如何解决某些特定的任务，而是学习到了学习本身这件事情，从而能提升自己在面对新任务上的学习能力。所以，Meta Learning 就是一门研究如何让机器学会更好地学习的新兴研究方向。

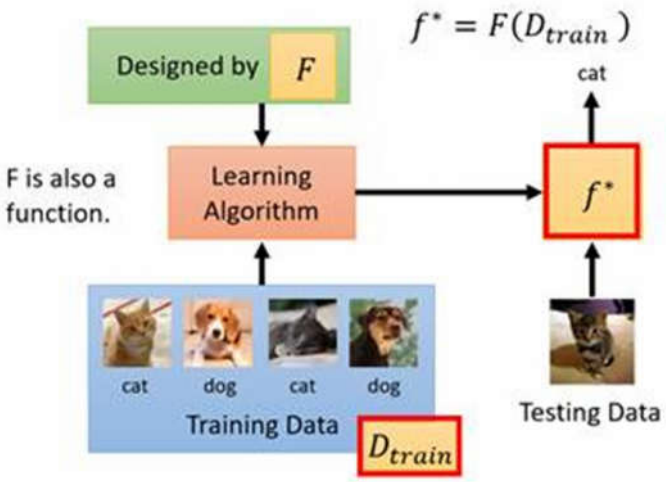


下面我们用具体的模型架构来解释一下 Meta Learning 实际上在做的事情。



首先，上图描述的是传统机器学习在做的事情——由人来设计一套学习算法，然后这个算法会输入一堆训练资料，通过长时间的训练得到算法里的参数，这堆参数拟合出一个函数 f^*，然后用测试资料来测试这个 f^*，如果效果达标就证明机器学到了该特定任务的实现函数 f^*。而 Meta Learning 做的事情与上述描述不同的地方在于，将其中由人来设计学习方法的过程，改成了由机器来设计一套学习方法。

如下图所示，如果将原本机器学习中的训练资料记为 D_{train}，那么在 Meta Learning 中的训练资料变为一堆 D_{train} 和一堆 f^* 的组合，然后现在机器要求解的结果不再是 f^*，而是一个新的函数 F：

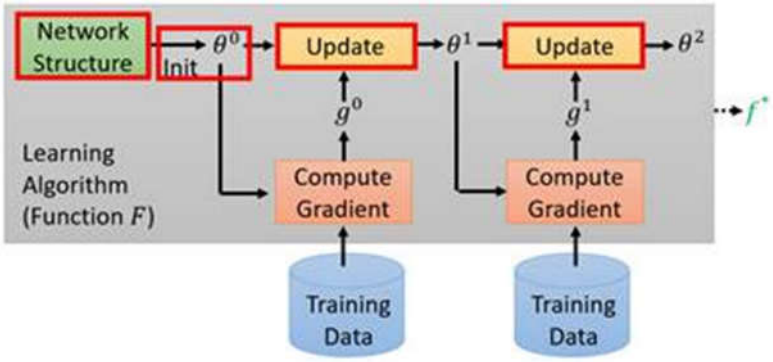


简言之，如果机器学习的定义表述为：根据资料找一个函数 f 的能力，如下图所示：

那么 Meta Learning 的定义就可以表述为：根据资料找一个函数 f 的函数 F 的能力，如下图所示：

现在，清楚了 Meta Learning 的架构搭建思路以后，我们就可以顺着该思路一步一步寻找解决方案。

首先第一步要做的，是准备 Meta Learning 的训练资料。前面说过，Meta Learning 的训练资料是一堆 D_{train} 和一堆 f^* 的组合，显然一堆 D_{train} 是很好准备的，于是重点在于，一堆 f^* 该如何准备。事实上，f^* 本身是一个抽象概念，我们需要知道它的具体实例是什么，不妨以传统的神经网络为例来介绍。



上图是大家都熟悉的梯度下降算法，它的流程可以简述为：设计一个网络架构->给参数初始化->读入训练数据批次->计算梯度->基于梯度更新参数->进入下一轮训练->……。对于每一个具体的任务来说，它的全部算法流程就构成了一个 *f**，也就是说，（如图中红色框架）每当我们采用了一个不同的网络架构，或使用了不同的参数初始化，或决定了不同的参数更新方式时，我们都在定义一个新的 *f**。所以，针对梯度下降算法来说，Meta Learning 的最终学习成果是在给定训练资料的条件下，机器能够找到针对这笔资料的 SGD 最佳训练流程（f_{best}*）。因此，前边我们探讨的为 Meta Learning 准备的 *f**，实际上是由包含尽量多和丰富的组合方式的不同训练流程来组成的。

3.3 准备知识

3.3.1 符号说明

Table 1: Notations

Notation	Definition or Descriptions
$\mathcal{U}, \mathcal{I}, C$	the user set, item set and scenario set
$m = \lvert\mathcal{U}\rvert, n = \lvert\mathcal{I}\rvert$	numbers of users and items
c	a recommendation scenario
$H_c \subset \mathcal{U} \times \mathcal{I}$	the interaction set of c
T_c	the learning task connected with c
$D_c^{\text{train}}, D_c^{\text{test}}$	the training set and testing set of T_c
$f_c : \mathcal{U} \times \mathcal{I} \to \mathbb{R}$	the recommender function for c
$\boldsymbol{\theta}_c$	parameters of f_c
$\boldsymbol{U}, \boldsymbol{I}$	embedding matrices for users and items
M	meta learner
$\mathbb{T}_{\text{meta-train}}, \mathbb{T}_{\text{meta-test}}$	meta-training set and meta-testing set
$\boldsymbol{\omega} = \{\boldsymbol{\omega}_R, \boldsymbol{\omega}_u, \boldsymbol{\omega}_s\}$	parameters of the meta learner
$\boldsymbol{\omega}_R$	shared initial parameters for f_c
$\boldsymbol{\omega}_u, \boldsymbol{\omega}_s$	parameters of update and stop controllers
$\boldsymbol{\alpha}, \boldsymbol{\beta}$	input gate and forget gate
$p^{(t)}$	stop probability at step t

我们的任务是对于每个用户 *u* 在场景 c 下推荐 top-n 个商品使得用户接下来的行为，即点击率和转化率的概率最大。

3.3.2 特定场景下的 Meta Learning 设定

在场景 c 下的推荐任务可以被认为是一个学习任务，它的训练集为 D_c^{train}。我们的目标是在给定 D_c^{train} 的情况下，学习一个元学习器 M，预测参数 f_c 和 θ_c。

假设已经获得一些列的训练任务作为 meta-training 集，定义为 $T_{meta\text{-}train}$。每个训练任务 $T_c \in T_{meta\text{-}train}$ 对应于一个场景 c，并且有它的训练集和测试集对：$D_c^{test} := \{(u,i,i^-)|(u,i) \in \boldsymbol{H}_c \wedge (u,i) \notin D_c^{train} \wedge (u,i^-) \notin \boldsymbol{H}_c\}$。为了得到元学习器的参数 ω，目标函数为：

$$\min_{\omega} \mathbb{E}_{T_c}\left[\mathcal{L}_{\omega}(D_c^{test}|D_c^{train})\right],$$

其中，

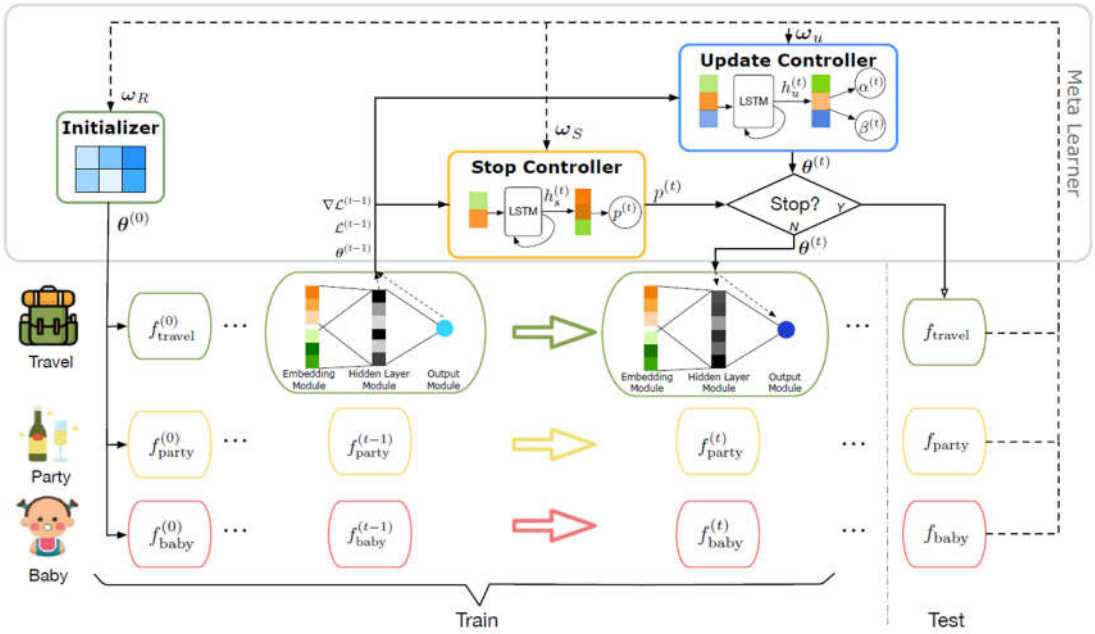
$$\mathcal{L}_{\omega}(D_c^{test}|D_c^{train}) = \sum_{(u_k,i_k,i_k^-)\in D_c^{test}} \frac{\ell(u_k,i_k,i_k^-;\theta_c)}{|D_c^{test}|},$$

$$\ell(u,i,i^-;\theta_c) = \max\left(0, \gamma - f(u,i;\theta_c) + f(u,i^-;\theta_c)\right),$$

γ=1，$T_{meta\text{-}train} \cap T_{meta\text{-}test} = \emptyset$。

3.4 算法原理

算法的整体框架如下图所示：



3.4.1 推荐网络

首先对用户 u 和商品 i 进行 one-hot 编码，$x_u \in \{0,1\}^m$ 和 $x_i \in \{0,1\}^n$，用户和商品 embedding 可以根据用户和商品的属性，或者没有上下文信息的交互来产生，采用的方法可以是协同过滤（collaborative filtering）或者网络嵌入（network embedding）的方法。所以：

$$\boldsymbol{e}_u = \boldsymbol{U}\boldsymbol{x}_u \quad \boldsymbol{e}_i = \boldsymbol{I}\boldsymbol{x}_i$$

隐藏模块（$e_u, e_i \rightarrow z_{ui}$）：将用户和商品的 embedding 连接起来，随后经过 L 个隐藏层得到用户和商品的连接表示：

$$z_{ui} = \phi_L(\cdots(\phi_1(\boldsymbol{e}_{ui}))\cdots)$$

$$\phi_l(\boldsymbol{z}) = \mathrm{ReLU}(\boldsymbol{W}_l\boldsymbol{z} + \boldsymbol{b}_l)$$

其中 W_l 和 b_l 是第 l 层的权值矩阵和偏置向量。

输出模块（$z_{ui}\rightarrow f_c(u,i)$）：

$$f_c(u,i) = \boldsymbol{w}^T z_{ui}$$

因此推荐参数 $\theta_c = \{(\boldsymbol{W}_l, \boldsymbol{b}_l)_{l=1}^L, \boldsymbol{w}\}$

3.4.2 s²Meta

主要为了解决以下三个问题：

1）如何初始化参数θ_c？

2）如何根据损失函数更新参数θ_c？

3）学习过程应该在何时停止？

具体的算法伪代码：

Algorithm 1 Training Algorithm of Meta Learner

Input: Meta-training set $\mathbb{T}_{\text{meta-train}}$, Loss function $\mathcal{L}$

1: $\omega_u, \omega_s, \omega_R \leftarrow$ Random Initialization
2: **for** $d \leftarrow 1, K$ **do** ▷ K is the number of meta-training steps
3: $D_c^{\text{train}}, D_c^{\text{test}} \leftarrow$ Random scenario from $\mathbb{T}_{\text{meta-train}}$
4: $\theta_c^{(0)} \leftarrow \omega_R, T \leftarrow 0$
5: **for** $t \leftarrow 1, T_{max}$ **do**
6: $B^{(t)} \leftarrow$ Random batch from D_c^{train}
7: $\mathcal{L}^{(t)} \leftarrow \mathcal{L}(B^{(t)}; \theta_c^{(t-1)})$
8: $p^{(t)} \leftarrow M_s(\mathcal{L}^{(t)}, ||\nabla_{\theta_c^{(t-1)}}\mathcal{L}^{(t)}||_2; \omega_s)$ ▷ Equation (9)
9: $s^{(t)} \sim \text{Bernoulli}(p^{(t)})$ ▷ Randomly decide to stop or not
10: **if** $s^{(t)} = 1$ **then**
11: Break;
12: **end if**
13: $\theta_c^{(t)} \leftarrow$ Update $\theta_c^{(t-1)}$ according to Equations (7) and (8)
14: $T \leftarrow T + 1$
15: **end for**
16: $\mathcal{L}^{\text{test}} \leftarrow \mathcal{L}(D_c^{\text{test}}; \theta_c^{(T)})$
17: Update ω_u, ω_R using $\nabla_{\omega_u}\mathcal{L}^{\text{test}}, \nabla_{\omega_R}\mathcal{L}^{\text{test}}$
18: $d\omega_s \leftarrow 0$
19: **for** $j \leftarrow 1, T$ **do**
20: $d\omega_s \leftarrow d\omega_s + (\mathcal{L}^{\text{test}} - \mathcal{L}(D_c^{\text{test}}; \theta_c^{(j)}))\nabla_{\omega_s}\ln(1 - p^{(j)})$
21: **end for**
22: Update ω_s using $d\omega_s$ ▷ Equation (11)
23: **end for**

ω_R 对应的是用户和商品映射到最终评分的 l 层的神经网络的参数 $\theta_c = \{(\boldsymbol{W}_l, \boldsymbol{b}_l)_{l=1}^L, \boldsymbol{w}\}$，每步开始时将 meta learner 的参数$\omega_R$赋值给$\theta_c$；$\omega_u$是更新器的参数；$\omega_s$是提早终止策略的参数；这三个参数会在每步当中进行更新，而每步当中会产生多个批次数据作为训练数据（第 6 行）来训练 l 层的神经网络的参数 θ_c。训练结束后，会根据θ_c在测试集的表现（第 16-22 行）来更新参数

$\omega = \{\omega_R, \omega_u, \omega_s\}$。

(1).参数初始化

对于第 3,4 行，每次从 $T_{meta\text{-}train}$ 选取的场景，要重新初始化这个场景 c 下的推荐网络的参数为$\theta_c^{(0)}$，$\theta_c^{(0)} = \omega_R$。$\omega_R$是 meta learner 的参数。

(2).更新策略

第 2 行，每一步 t 下获得一个 batch 的数据：$B^{(t)} = \{u_k, i_k, i_k^-\}_{k=1}^N$，根据损失函数进行更新：

$$\mathcal{L}^{(t)} = \sum_{(u_k, i_k, i_k^-) \in B^{(t)}} \frac{\ell(u_k, i_k, i_k^-; \theta_c^{(t-1)})}{|B^{(t)}|}$$

通常，SGD 采用的更新函数为：

$$\theta_c^{(t)} = \theta_c^{(t-1)} - \alpha \nabla_{\theta_c^{(t-1)}} \mathcal{L}^{(t)}$$

然而在小样本学习中，不合适的学习率会使得函数陷入局部最优，因此，这里设计了更新器去更新参数，θ_c的更新策略为：

$$\theta_c^{(t)} = \boldsymbol{\beta}^{(t)} \odot \theta_c^{(t-1)} - \boldsymbol{\alpha}^{(t)} \odot \nabla_{\theta_c^{(t-1)}} \mathcal{L}^{(t)}$$

这里参数$\boldsymbol{\alpha}^{(t)}$是输入门，类似学习率；$\boldsymbol{\beta}^{(t)}$是遗忘门，为了帮助跳出局部最优。这里采用了 LSTM 来更新这两个参数：

$$\begin{aligned}
\boldsymbol{h}_u^{(t)}, \boldsymbol{c}_u^{(t)} &= \text{LSTM}([\nabla_{\theta_c^{(t-1)}} \mathcal{L}^{(t)}, \mathcal{L}^{(t)}, \theta_c^{(t-1)}], \boldsymbol{h}_u^{(t-1)}, \boldsymbol{c}_u^{(t-1)}), \\
\boldsymbol{\beta}^{(t)} &= \sigma(\boldsymbol{W}_F \boldsymbol{h}_u^{(t)} + \boldsymbol{b}_F), \\
\boldsymbol{\alpha}^{(t)} &= \sigma(\boldsymbol{W}_I \boldsymbol{h}_u^{(t)} + \boldsymbol{b}_I),
\end{aligned}$$

$\omega_u = \{\boldsymbol{W}_F, \boldsymbol{b}_F, \boldsymbol{W}_I, \boldsymbol{b}_I\}$。

(3).提早终止策略

为了防止过拟合，设置了提早终止策略，通过停止控制器 M_s 来控制，提前终止的概率为 p。这里仍然采用 LSTM 来计算概率 p：

$$\begin{aligned}
\boldsymbol{h}_s^{(t)}, \boldsymbol{c}_s^{(t)} &= \text{LSTM}([\mathcal{L}^{(t)}, ||\nabla_{\theta_c^{(t-1)}} \mathcal{L}^{(t)}||_2], \boldsymbol{h}_s^{(t-1)}, \boldsymbol{c}_s^{(t-1)}) \\
p^{(t)} &= \sigma(\boldsymbol{W}_s \boldsymbol{h}_s^{(t)} + b_s),
\end{aligned}$$

$\omega_s = \{\boldsymbol{W}_s, \boldsymbol{b}_s\}$。

(4).训练 meta learner 参数

meta learner 的目标是在测试集中的损失函数达到最小，ω_R和ω_u关于元测试损失函数$\mathcal{L}_\omega(D_c^{\text{test}} | D_c^{\text{train}})$的梯度可以通过反向传播来计算。而$\omega_s$需要采用随机策略梯度来优化。这里定义了每一步 t 的回报，用一步更新后的损失来衡量：

$$r^{(t)} = \mathcal{L}(D_c^{\text{test}}; \theta_c^{(t-1)}) - \mathcal{L}(D_c^{\text{test}}; \theta_c^{(t)})$$

那么在 t 步累计的回报为：

$$Q^{(t)} = \sum_{i=t}^{T} r^{(t)} = \mathcal{L}(D^{\text{test}}; \theta_c^{(t-1)}) - \mathcal{L}(D^{\text{test}}; \theta_c^{(T)})$$

根据强化学习策略，按照下式更新ω_s：

$$\omega_S \leftarrow \omega_S + \gamma \sum_{t=1}^{T} Q^{(t)} \nabla_{\omega_S} \ln M_S(\mathcal{L}^{(t)}, ||\nabla_{\theta_c^{(t-1)}} \mathcal{L}^{(t)}||_2; \omega_S)$$

3.5 算法应用

数据集如下图所示，最后一列表示用于少样本学习的场景数

Table 2: Statistics of the Datasets. #Inter. denotes the number of user-item interactions and #Scen. denotes the number of scenarios we use as few-shot tasks.

Dataset	#Users	#Items	#Inter.	#Scen.
Amazon	766,337	492,505	17,523,124	1,289
Movielens	138,493	27,278	20,000,263	306
Taobao	775,603	1,452,525	5,717,835	355

性能结果如图所示

Table 3: The top-N recall results on test scenarios.

Method	Amazon			Movielens			Taobao		
	Recall@10	Recall@20	Recall@50	Recall@10	Recall@20	Recall@50	Recall@20	Recall@50	Recall@100
NeuMF	24.55	35.65	55.19	31.67	51.30	84.98	25.64	42.31	58.84
ItemPop	26.86	32.65	50.42	39.65	54.32	78.12	18.25	28.57	32.44
CDCF	10.27	16.72	32.79	29.19	41.04	65.75	9.81	20.93	32.14
CMF	27.64	38.31	55.24	29.80	46.91	74.37	9.16	16.47	29.31
EMCDR	31.71	42.14	58.73	43.55	60.89	83.54	20.43	31.52	45.67
CoNet	30.17	41.57	56.06	46.62	63.61	87.06	20.27	31.48	44.53
s^2Meta	**34.39**	**46.53**	**64.28**	**47.79**	**66.02**	**89.07**	**27.11**	**44.10**	**59.98**
Improve	8.35	10.42	9.45	2.51	3.79	2.31	5.73	4.23	1.90

4. HERS: Modeling Influential Contexts with Heterogeneous Relations for Sparse and Cold-Start Recommendation

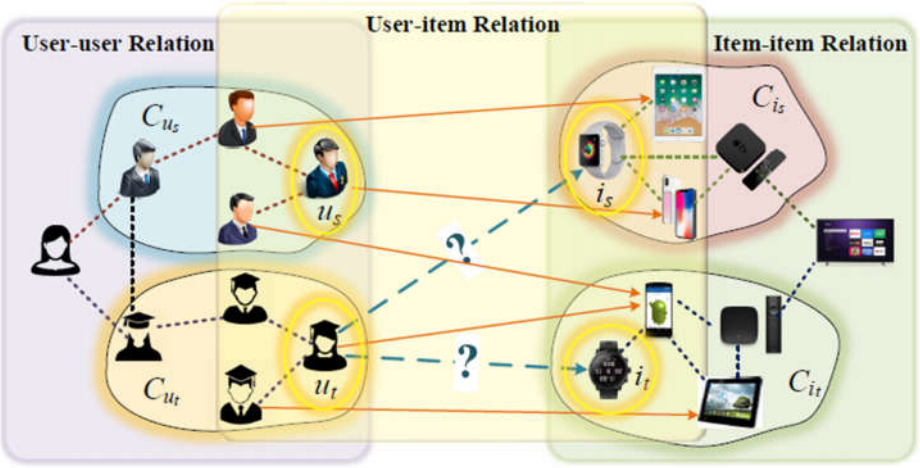
AAAI, 2019

HERS: Modeling Influential Contexts with Heterogeneous Relations for Sparse and Cold-Start Recommendation

Liang Hu,[1,2] Songlei Jian,[1,3] Longbing Cao,[1] Zhiping Gu,[4] Qingkui Chen,[2] Artak Amirbekyan[5]
[1]Advanced Analytics Institute, University of Technology Sydney, Australia
[2]University of Shanghai for Science and Technology, China, [3]National University of Defense Technology, China
[4]Shanghai Technical Institute of Electronics Information, China, [5]Commonwealth Bank of Australia
rainmilk@gmail.com, jiansonglei@163.com, longbing.cao@uts.edu.au
guzhiping@stiei.edu.cn, chenqingkui@usst.edu.cn, artak.amirbekyan@cba.com.au

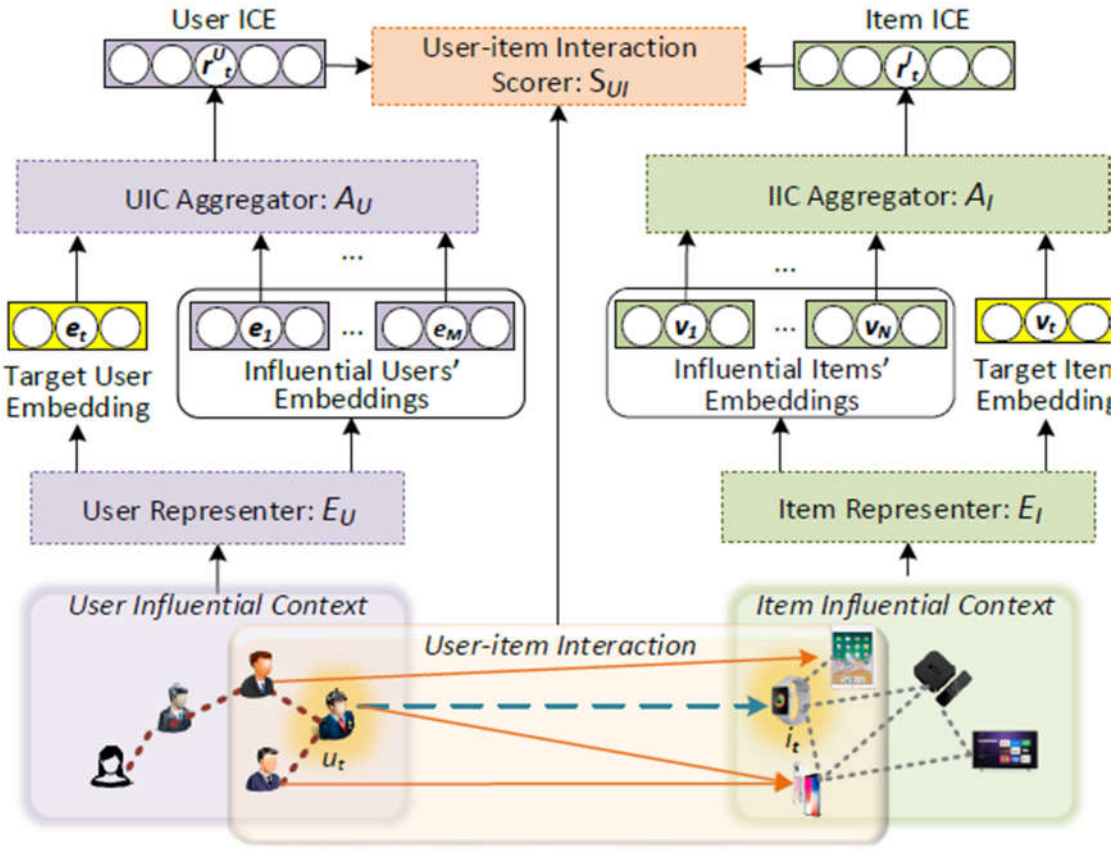
4.1 算法背景

给定一个用户 user，他的选择也会受到社交网络中朋友的影响。同样，用户选择的商品也会受到它相关商品的影响。通过分析用户-用户之间的关系和商品-商品之间的关系，可以进一步解决用户-商品交互的稀疏性以及用户和商品冷启动问题。



4.2 Neural ICAU-based HERS Model

4.2.1 模型架构



模型架构主要包括5部分：User 表示 EU, UIC（User Influential Context） 聚合器 AU, Item 表示 EI , IIC（Item Influential Context） 聚合器 AI 和 user-item 交互得分 SUI：

- User Representer E_U: it maps target user u_t and its influential users in UIC to the corresponding user embeddings, i.e., $E_U(\mathcal{U}_{u_t}) \mapsto \mathcal{E}_{u_t}$ where $\mathcal{E}_{u_t} = \{\mathbf{e}_t, \mathbf{e}_1, \cdots \mathbf{e}_M\}$.
- Item Representer E_I: it maps target item i_t and its influential items in IIC to the corresponding item embeddings, i.e., $E_I(\mathcal{I}_{i_t}) \mapsto \mathcal{E}_{i_t}$ where $\mathcal{E}_{i_t} = \{\mathbf{v}_t, \mathbf{v}_1, \cdots \mathbf{v}_N\}$.
- UIC Aggregator A_U: it learns a representation $\mathbf{r}_t^U$ for the influential context $\mathcal{C}_{u_t}$, namely influential context embedding (ICE). Formally, we have $A_U(\mathcal{C}_{u_t}, \mathcal{E}_{u_t}) \mapsto \mathbf{r}_t^U$.
- IIC Aggregator A_I: it learns i_t's ICE by aggregating the influential context $\mathcal{C}_{i_t}$, that is, $A_I(\mathcal{C}_{i_t}, \mathcal{E}_{i_t}) \mapsto \mathbf{r}_t^I$.
- User-item Interaction Scorer S_{UI}: it learns to score the interaction strength between the target user-item pair $\langle u_t, i_t \rangle$ in terms of the user ICE $\mathbf{r}_t^U$ and the item ICE $\mathbf{r}_t^I$, namely $S_{UI}(\mathbf{r}_t^U, \mathbf{r}_t^I, y_{u_t,i_t}) \mapsto s_{\langle \mathcal{C}_u, \mathcal{C}_i \rangle}$ (cf. Eq. 1).

4.2.2 Influential-context 聚合单元

Influential-context 聚合单元（Influential-context Aggregation Unit）目的是根

据用户的上下文用户得到用户的表示$\mathcal{E}_{u_t}$，根据商品的上下文关系得到商品的表示$\mathcal{E}_{u_t}$。聚合过程包括两步：S1 和 S2。

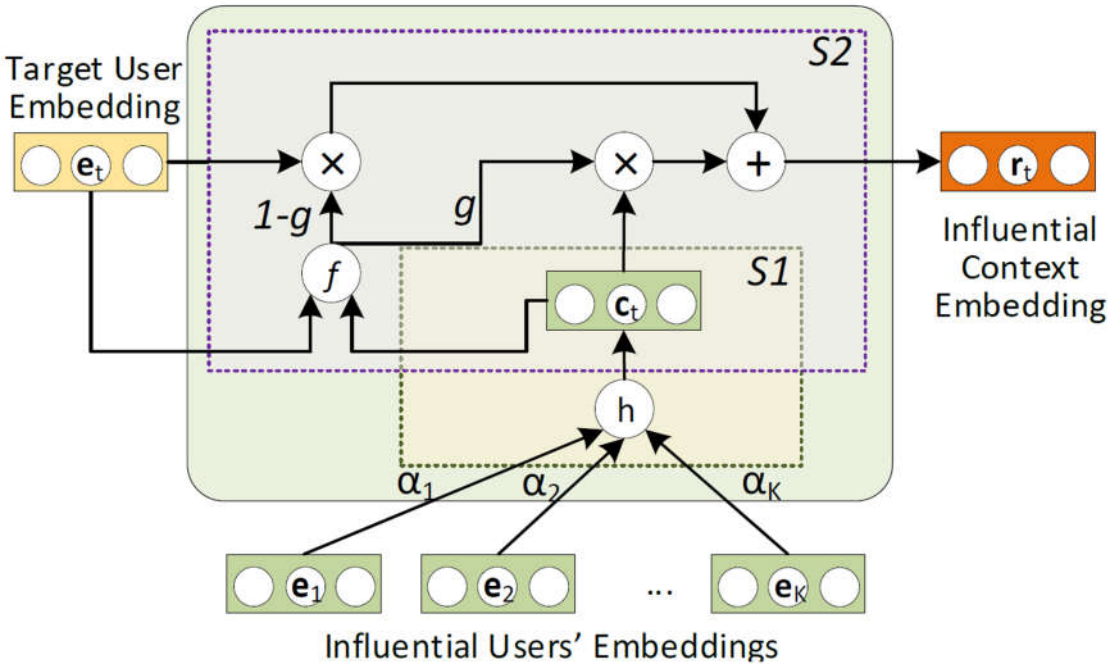
S1：该过程输出的是辅助影响 embedding c_t：

$$\{\alpha_1, \cdots \alpha_K\} = a(\mathbf{e}_1, \cdots \mathbf{e}_K)$$
$$\mathbf{c}_t = h(\mathbf{e}_1, \cdots \mathbf{e}_K | \alpha_1, \cdots \alpha_K)$$

$e_1,\ldots,e_K$ 是目标用户 e_t 的上下文用户，通过 a()这个函数得到每个用户的权重系数α1, ..., αK。随后通过 h()这个函数将上下文用户的 embedding 融合成辅助 embedding c_t。

S2：将目标用户的 embedding e_t 和辅助 embedding c_t 融合成最终的用户表示 r_t：

$$g = f(\mathbf{c}_t, \mathbf{e}_t)$$
$$\mathbf{r}_t = g\mathbf{c}_t + (1-g)\mathbf{e}_t$$



4.2.3 User's Influential Context Embedding

给定用户有影响力的上下文$\mathcal{C}_{u_t} = \{\mathcal{U}_{u_t}, \mathcal{R}_{u_t}\}$, $\mathcal{U}_{u_t}$包括了目标用户 u_t 和它的一阶上下文邻居$\{u_{t,m}\}_{1\le m\le M}$和每个一阶用户 $u_{t,m}$（embedding 表示为 $e_{t,m}$）的邻居$\{u_{t,m,k}\}_{1\le k\le K_m}$（embedding 表示为 $e_{t,m,k}$）,也就是 u_t 的二阶上下文邻居。

二阶 ICAUs：

S1：为了获得辅助 embedding，首先将二阶上下文用户通应到一个 tanh 层：

$$\mathbf{h}_{t,m,k} = \tanh(\mathbf{W}^{(1)}\mathbf{e}_{t,m,k} + \mathbf{b})$$

随后得到该用户的权重系数：

$$\alpha^{U}_{t,m,k} = \mathrm{softmax}\Big(isr_{\theta}\big(\mathbf{W}^{(2)}\mathbf{h}_{t,m,k}\big)\Big)$$

其中：

$$\mathrm{softmax}(x_k) = e^{x_k} / \sum_j e^{x_j}$$

$$isr_{\theta}(x) = \frac{x}{\sqrt{1+\theta x^2}}$$

最后，得到辅助影响 embedding $c_{t,m}$：

$$\mathbf{c}_{t,m} = \sum_{k=1}^{K_m} \alpha^{U}_{t,m,k}\mathbf{e}_{t,m,k}$$

S2：根据得到的辅助影响 embedding，和目标用户的一阶用户 $u_{t,m}$ 的 embedding 融合：

$$\mathbf{r}_{t,m} = g_{t,m}\mathbf{c}_{t,m} + (1 - g_{t,m})\mathbf{e}_{t,m}$$

其中：

$$g_{t,m} = \sigma\Big(isr_{\theta}\big(\mathbf{W}^{(3)}\tanh\left(\mathbf{W}^{(4)}\mathbf{c}_{t,m} + \mathbf{W}^{(5)}\mathbf{e}_{t,m}\right)\big)\Big)$$

$$\sigma(z) = 1/(1+e^{-z})$$

一阶 ICAUs：

根据上一步得到的每个一阶用户的 embedding $r_{t,m}$ 得到该用户 u_t 的辅助 embedding：

$$\mathbf{c}^{U}_{t} = \sum_{m=1}^{M} \alpha^{U}_{t,m}\mathbf{r}_{t,m}$$

其中的$\alpha_{t,m}$和二阶 ICAUs 中的计算方式相同。

随后根据辅助 embedding 和用户自身的 embedding 进行融合，得到最终的 embedding 表示：

$$\mathbf{r}^{U}_{t} = g^{U}_{t}\,\mathbf{c}^{U}_{t} + (1 - g^{U}_{t})\mathbf{e}_{t}$$

其中 g 函数和二阶 ICAUs 中的计算方式相同。

4.2.4 Item's Influential Context Embedding

和用户考虑了二阶用户不同，这里的商品只考虑一阶商品的交互。首先得到该商品的辅助 embedding：

$$\mathbf{c}^{I}_{t} = \sum_{n=1}^{N} \alpha^{I}_{t,n}\mathbf{v}_{t,n}$$

其中的权重系数$\alpha_{t,n}$可以按照下式获得：

$$\alpha_{t,n}^{I} = \mathrm{softmax}\Big(isr_{\theta}\big(\mathbf{W}^{(6)}\tanh\left(\mathbf{W}^{(7)}\mathbf{v}_{t,n}\right)\big)\Big)$$

随后根据辅助 embedding 和商品自身的 embedding 进行融合：

$$\mathbf{r}_{t}^{I} = g_{t}^{I}\mathbf{c}_{t}^{I} + (1 - g_{t}^{I})\mathbf{v}_{t}$$

其中

$$g_{t}^{I} = \sigma\Big(isr_{\theta}\big(\mathbf{W}^{(8)}\tanh\left(\mathbf{W}^{(9)}\mathbf{c}_{t}^{I} + \mathbf{W}^{(10)}\mathbf{v}_{t}\right)\big)\Big)$$

4.2.5 User-item Interaction Ranking

用户和商品之间的评分按照内积进行计算：

$$S_{\langle u_t, i_t\rangle} = \mathbf{r}_{t}^{U\top}\mathbf{r}_{t}^{I}$$

用户和商品如果有过交互，认为是正项的商品 i_p；如果没有交互，认为是负项的商品 i_n，因此可以认为：

$$S_{\langle u_t, i_p\rangle} \geq S_{\langle u_t, i_n\rangle}$$

这里采用最大边界损失函数（max-margin loss）：

$$L_{\langle u_t, i_p\rangle \succeq \langle u_t, i_n\rangle} = \max\{0, m - S_{\langle u_t, i_p\rangle} + S_{\langle u_t, i_n\rangle}\}$$

最终的目标函数是：

$$\mathcal{L} = \frac{1}{|\mathcal{B}|}\sum_{\langle u_t, i_p, i_n\rangle \in \mathcal{B}} L_{\langle u_t, i_p\rangle \succeq \langle u_t, i_n\rangle}$$