

深入理解推荐系统：召回

原创 Coggle Coggle数据科学 4月2日

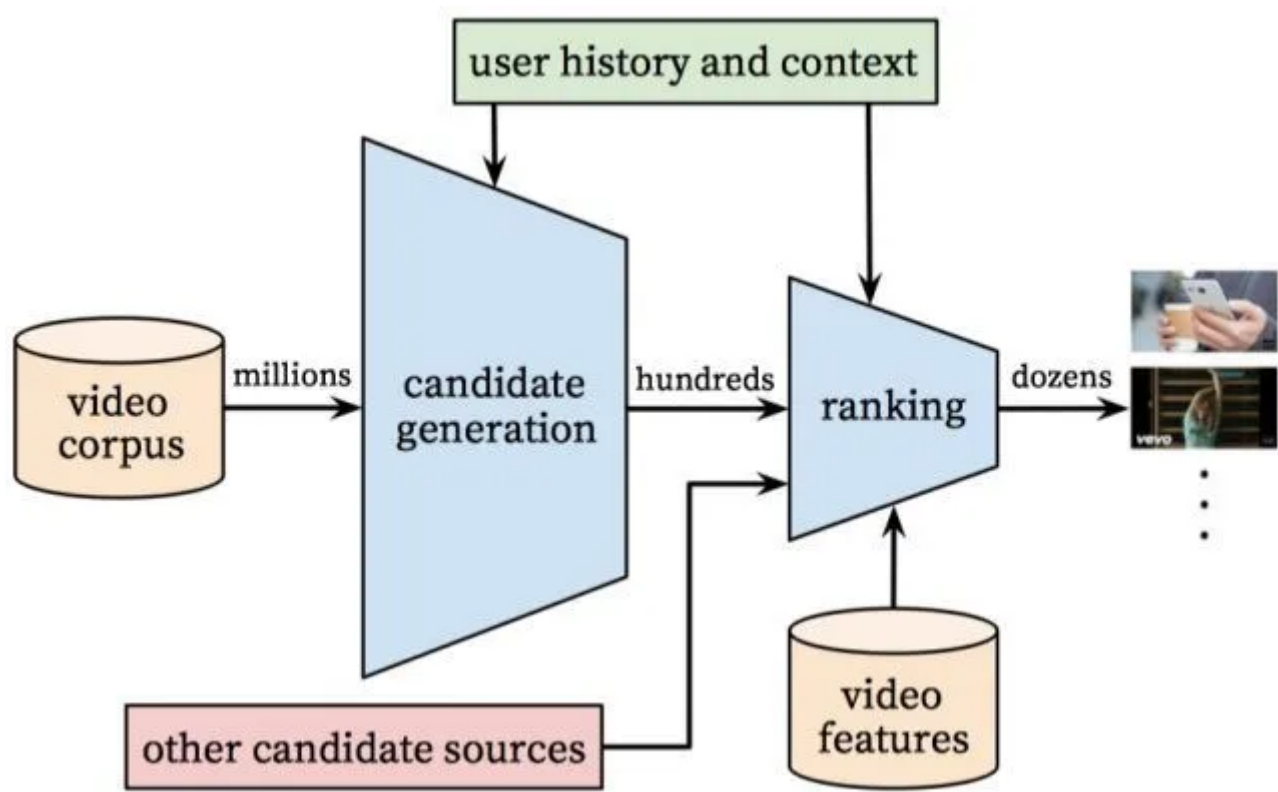
写在前面

【推荐系统】专栏历史文章：

深入理解YouTube推荐系统算法：<https://zhuanlan.zhihu.com/p/114703091>

作为【推荐系统】系列文章的第二篇，将以“召回”作为今天的主角，会从四个方面来介绍召回的不同算法方式，即基于内容的召回、协同过滤、基于FM模型召回和基于深度学习的方法。

一、背景介绍



推荐系统整体架构

召回是推荐系统的第一阶段，主要根据用户和商品部分特征，从海量的物品库里，快速找回一小部分用户潜在感兴趣的物品，然后交给排序环节。这部分需要处理的数据量非常大，速度要求快，所有使用的策略、模型和特征都不能太复杂。下面主要介绍四种常见的召回方法：

- **基于内容的召回**：使用item之间的相似性来推荐与用户喜欢的item相似的item。

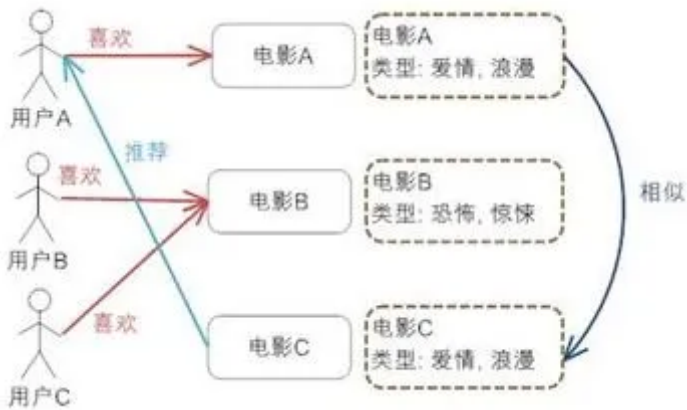
例如：如果用户A看了《绣春刀2》这部杨幂主演的电影后，则会为他推荐杨幂主演的其他电影或电视剧

- **协同过滤**：同时使用query和item之间的相似性来进行推荐。

例如：如果用户A与用户B类似，并且用户B喜欢视频1，则系统可以向用户A推荐视频1（即使用户A尚未看过任何与视频1类似的视频）。

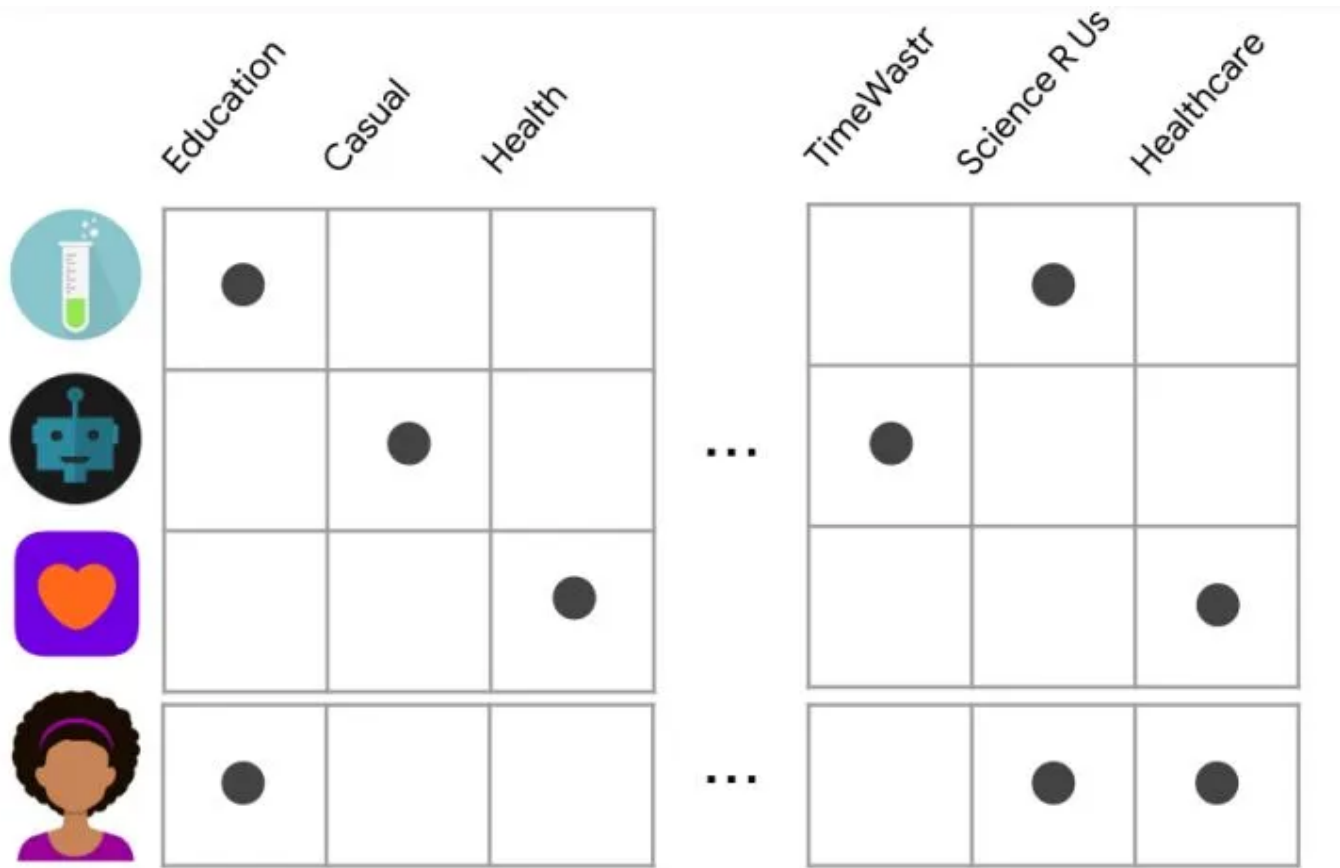
- **基于FM模型召回：**FM是基于矩阵分解的推荐算法，其核心是二阶特征组合。
- **基于神经网络的方法：**利用神经网络生成相应的候选集。

二、基于内容的召回



基于内容的召回（CB召回），一般也叫做标签召回。当谈起CB的时候，大家可能会觉的很简单，用tag或者用cate召回就行了，好像没什么可做的。可事实上，CB并不仅仅是用tag和cate做个倒排就搞定了。这类召回的核心思想是基于item自身的属性，这些属性可以表达为tag，Cate，也可以用来表达用户ID，用户类型等，更可以通过一些交叉验证的方式，针对内容提取向量，将内容表达为连续向量的方式进行召回。接下来我们进一步来理解基于内容的过滤。

在实际的应用中，如电影推荐，首先我们根据用户之前的历史行为信息（如点击，评论，观看等），CB会使用item相关特征来推荐给用户与之前喜欢的item类似的item。为了更形象的表示CB，假设一个应用商店要推荐给用户相应的APP。下图是相应的特征矩阵，其中每一行代表一个应用程序，每一列代表一个特征。包括不同的类别特征，应用程序的发布者信息等。为简化起见，假定此特征矩阵是布尔类型的：非零值表示应用程序具有该特征。



还可以在同一特征空间中表示用户。一些与用户相关的特征可以由用户明确提供。例如，用户在其个人资料中选择“娱乐应用”。其他特征可能是隐式的，具体取决于它们先前安装的应用程序。例如，用户安装了由Science R Us发布的另一个应用程序。

模型应推荐与此用户有关的item。为此，必须首先选择一个相似性指标（如，点积）。然后，推荐系统会根据此相似性度量标准为每个候选item打分。请注意，建议是针对该用户的，因为该模型未使用其他用户的任何信息。

2.1 基于内容召回优缺点

优点

- 该模型不需要其他用户的任何数据，因为推荐是针对该用户的。这使得更容易扩展到大量用户。
- 该模型可以捕获用户的特定兴趣。

缺点

- 由于item的特征表示在某种程度上是手工设计的，因此该技术需要大量领域知识。因此，模型很依赖手工设计特征的好坏。
- 该模型只能根据用户的现有兴趣提出建议。换句话说，该模型扩展用户现有兴趣的能力有限。

基于内容的召回看似比较容易，如果当我们的item属性越来越多的时候，比如一个视频可能有多个平行的tag以及其它属性，那么，为了把这些信息综合利用起来，我们还会利用多term检索的方式，去提升CB的效果。下面，我们针对这些内容详细的来聊一聊CB的常见优化点。

2.2 倒排优化

优化倒排的主要目的是提升cb召回的推荐效果，常见的倒排基本是和线上排序指标一致的，比如，如果排序的指标是点击率，倒排理所应当，也是点击率。但这样的排序方式有个小问题，因为倒排排序使用的是后验的值，而排序通常也是单指标排序，这样，就容易出现我们之前提到的，单指标被hack的问题，比如，用点击率倒排，头部都是标题党等。所以，这个问题是需要额外注意的。另外，也要考虑指标的bias的问题，例如，用完成率倒排导致短的视频都排到了头部。这种问题可以通过归一化的方式缓解。但有一个潜在风险，资源的后验表现的分布往往会跟资源本身的类型有关。

2.3 触发key的优化

key的优化要求只有一点，保证每次选择的key，是用户点击概率最大的key即可，所以通用的方式是把用户的点击历史按照属性加和取top，比如，在某个类别上点击的次数排序，把点击次数最多的那几个类别留作触发的key，这个过程很简单，没太多优化点，我们就不继续讨论了。这里想聊的是关于用户不一样的行为差异化权重的问题，我们选取key的过程实际是判断用户对某一类内容感兴趣的过程，也就是通过行为，来判断用户的感兴趣程度，在只有浏览功能的产品里，点击就是表达用户兴趣的唯一行为，但产品通常会设计很多交互功能，来帮助用户进行有效的兴趣表达，所以，在触发key的选取的时候需要考虑到这一点，至于怎么做是和业务形态相关的事，这里就不展开了。

2.4 多维度内容属性

最后，我们来讲一下cb里面比较高阶的问题，多term的问题。我们先考虑这样一种情况，一个视频有很多个tag，tag的特性是平行不唯一。我们在线通过点击量汇聚得到了用户的高频点击tag，通常的方式是，每一个tag都会有一个倒排，然后各自召回。但很容易想到的，当用户同时有“帅哥”，“萌宠”这两个tag的时候，通过萌宠给用户召回一个美女+萌宠的视频显然没有召回一个帅哥+萌宠的视频更有吸引力，也就是说，我们在召回的时候，如果能同时考虑多个term之间的关系，会更有效一些。传统搜索里对于多term召回是有一套实现方式的，通过设置每个term的权重，可以在返回结果里去得到包含多term的结果，这部分属于搜索架构的内容范畴，我在此不展开了，有兴趣的同学可以找身边做搜索的同学了解一下。我们讲一个更和推荐match的方式，用户身上有多个标签，内容上面也有多个标签，我们在做多term匹配的时候，就是标签的list到标签的list，使得他们点击的概率最大即可。是不是觉得豁然开朗了？是不是转化成点击率建模问题了？更简单地，直接使用word2vec，把用户标签和内容标签作为一个sentence训练，再离线把内容的标签加和表征为内容的属性向量，在线做召回即可。当然，也可以用更复杂的方式来做，提升精度，大同小异，就不展开说了。那作者，内容的其他纬度等信息也是可以一样的方式加进去的，这就留给大家来讨论了。

三、协同过滤

为了解决基于内容的召回所存在的弊端，人们提出了协同过滤召回方式（Collaborative filtering, CF），CF同时使用user和item之间的相似性来进行推荐。这样可以提高模型的推荐拓展性。也就是说，协同过滤模型可以根据相似用户B的兴趣向用户A推荐商品。此外，可以自动学习Embedding，而无需依赖手工设计的特征。

一般来说，协同过滤推荐分为三种类型。第一种是**基于用户(user-based)的协同过滤**，第二种是**基于项目(item-based)的协同过滤**，第三种是**基于模型(model based)的协同过滤**。

- **基于用户(user-based)的协同过滤**：主要考虑的是用户和用户之间的相似度，只要找出相似用户喜欢的物品，并预测目标用户对对应物品的评分，就可以找到评分最高的若干个物品推荐给用户。
- **基于项目(item-based)的协同过滤**：和基于用户的协同过滤类似，只不过这时我们转向找到物品和物品之间的相似度，只有找到了目标用户对某些物品的评分，那么我们就可以对相似度高的类似物品进行预测，将评分最高的若干个相似物品推荐给用户。比如你在网上买了一本机器学习相关的书，网站马上会推荐一堆机器学习，大数据相关的书给你，这里就明显用到了基于项目的协同过滤思想。
- **基于模型 (model based) 的协同过滤**：是目前最主流的协同过滤类型了，所含算法是非常之多的，如矩阵分解、关联算法、聚类算法、深度学习、图模型等等。

这里我们带来一个有关电影推荐系统的简单例子帮助更好的理解协同过滤。首先，考虑一个电影推荐系统，其中训练数据由一个反馈矩阵组成，其中每行代表一个user，每一列代表一个item。

关于电影的反馈分为以下两类：

- **显示反馈**：用户通过提供评分来指定他们对特定电影的喜欢程度。
- **隐式反馈**：如果用户观看电影，则系统会推断用户感兴趣。

假设反馈矩阵是布尔类型的，即值为1和0分别表示对电影是否感兴趣，当用户访问首页时，系统会根据以下两种情况推荐电影：

- 与用户过去喜欢的电影相似 (Item-Based CF)
- 类似用户喜欢的电影 (User-Based CF)

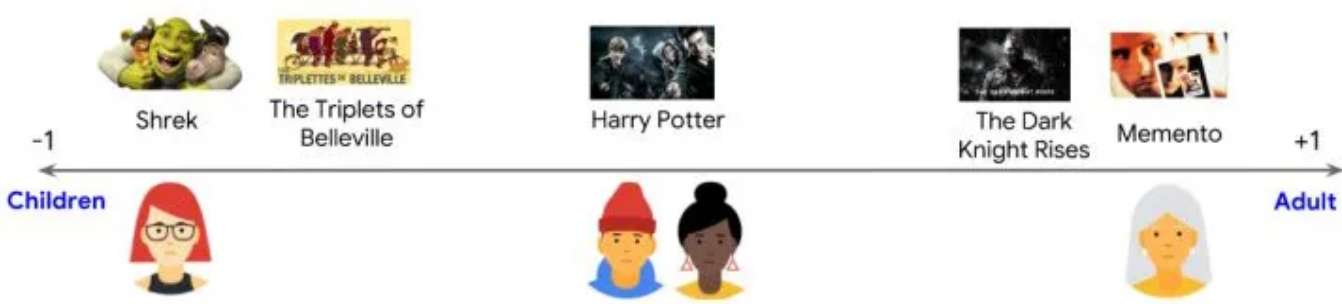
为便于举例阐述，让我们手工设计用以描述电影的一些特征：

Movie	Rating	Description
The Dark Knight Rises	PG-13	Batman endeavors to save Gotham City from nuclear annihilation in this sequel to The Dark Knight , set in the DC Comics universe.
Harry Potter and the Sorcerer's Stone	PG	A orphaned boy discovers he is a wizard and enrolls in Hogwarts School of Witchcraft and Wizardry, where he wages his first battle against the evil Lord Voldemort.
Shrek	PG	A lovable ogre and his donkey sidekick set off on a mission to rescue Princess Fiona, who is imprisoned in her castle by a dragon.
The Triplets of Belleville	PG-13	When professional cyclist Champion is kidnapped during the Tour de France, his grandmother and overweight dog journey overseas to rescue him, with the help of a trio of elderly jazz singers.
Memento	R	An amnesiac desperately seeks to solve his wife's murder by tattooing clues onto his body.

电影描述

3.1 一维Embedding

假设我们为每部电影分配一个标量，用于描述该电影是适合儿童（负值）还是适合成人（正值）观看。假设我们还为每个用户分配了一个标量，用于描述用户对儿童电影（接近-1）或成人电影（接近+1）的兴趣。对于我们希望用户喜欢的电影，电影Embedding和用户Embedding的乘积应更高（接近1）。

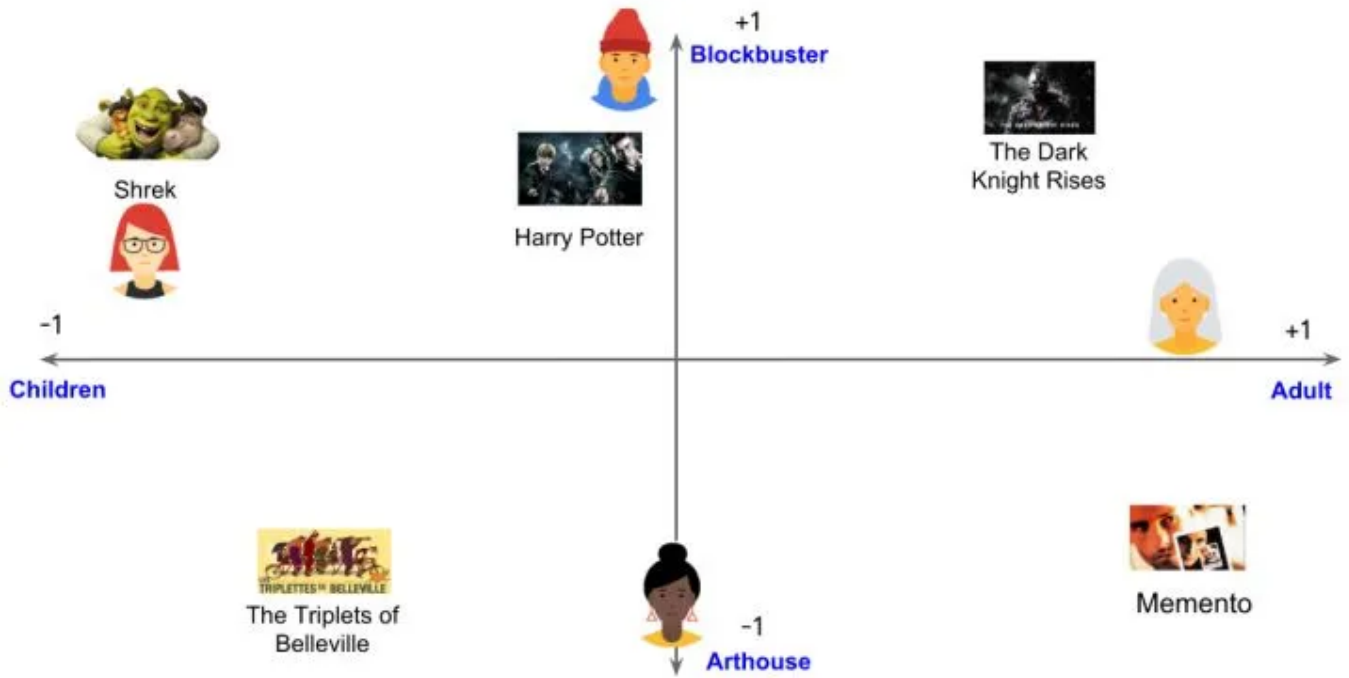


在下图中，每个对号标记都标识特定用户观看的电影。第三和第四用户具有的特征很好地表示了用户的偏好：第三用户偏爱儿童电影，第四用户偏爱成人电影。但是，单个特征无法很好地表示第一和第二用户的偏好。这时候需要考虑二维甚至更高维度的特征。

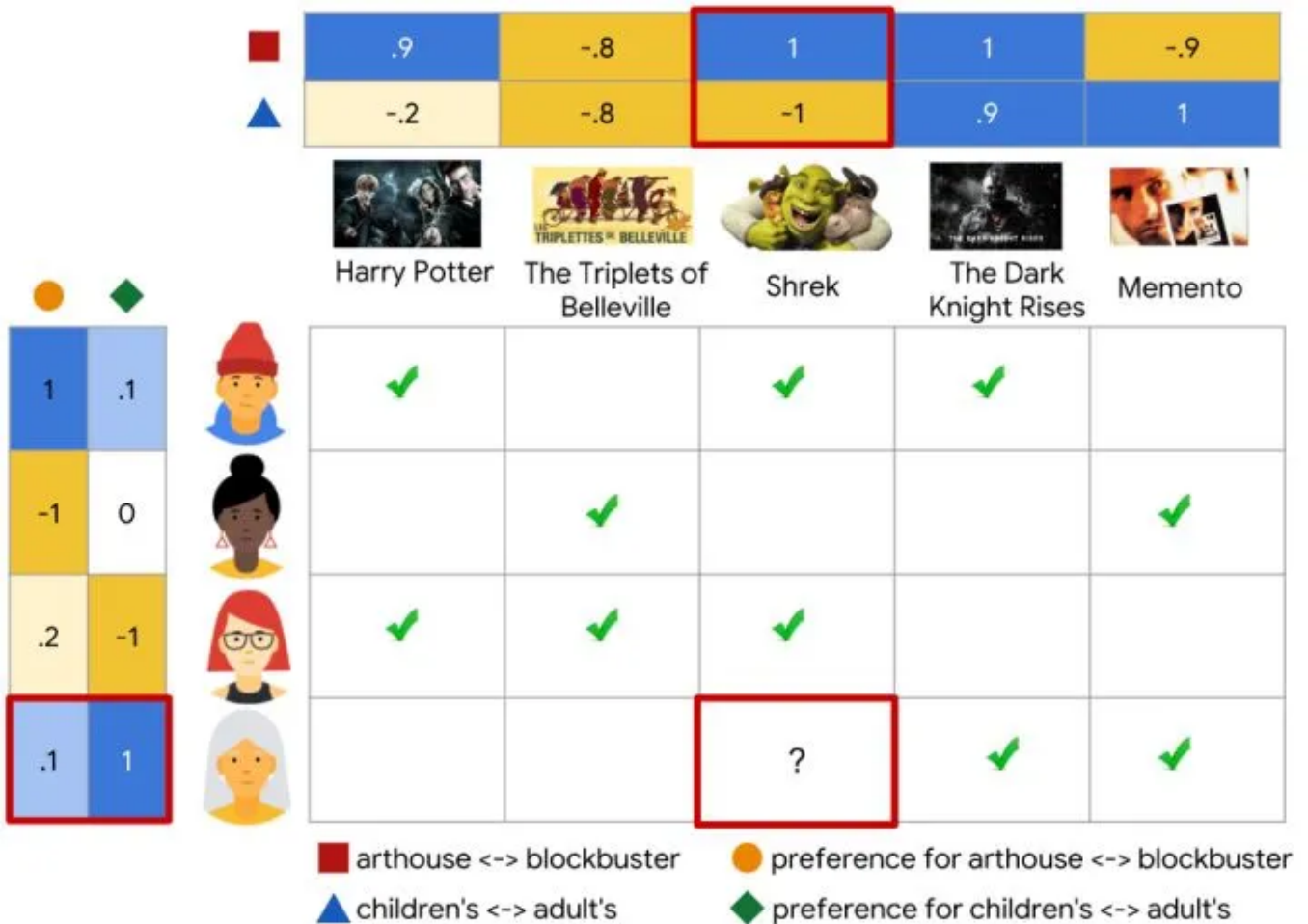


3.2 二维Embedding

一个特征不足以解释所有用户的偏好。为了克服这个问题，让我们添加第二个特征：每部电影是商业流行片或是小众文艺片上的表现程度。通过这个特征，我们现在可以使用以下二维Embedding来表示每部电影：



我们再次将用户放置在相同的嵌入空间中，以最好地解释反馈矩阵：对于（用户，商品）对，我们希望用户Embedding和商品Embedding的点积在用户观看商品时接近1 电影，否则为0。



在这个例子中，我们对Embedding进行了手工设计。在实践中，可以自动学习Embedding向量表示，这是协同过滤模型的强大功能。在接下来的内容中，我们将讨论学习这些嵌入表示的不同模型以及如何对其进行训练。

当模型自动学习Embedding时，这种方法的协同性质就显而易见了。假设电影的Embedding向量是固定的。然后，模型可以为用户学习Embedding向量，以最好地解释他们的偏好。因此，具有相似偏好的用户的Embedding将紧密在一起。同样，如果用户的Embedding是固定的，则我们可以学习电影Embedding以最好地解释反馈矩阵。结果，类似用户喜欢的电影的Embedding将在Embedding空间中紧密在一起。

3.3 基于模型的协同过滤

矩阵分解是一个简单的Embedding模型。给定反馈矩阵 $A \in \mathbb{R}^{m \times n}$ ，其中 m 是用户（或 query）数量， n 是item数量，该模型将学习：

- user Embedding矩阵 $U \in \mathbb{R}^{m \times d}$ ，其中第*i*行是 $user_i$ 的Embedding。
- tem Embedding矩阵 $V \in \mathbb{R}^{n \times d}$ ，其中第*j*行是 $item_j$ 的Embedding。



E mbedding通过学习，使得 UV^T 的乘积是反馈矩阵 A 的良好近似。 UV^T 的 (i,j) 项就是 $user_i$ 和 $item_j$ 对应的两个embedding的点积，使其尽量接近 $A_{i,j}$ 。

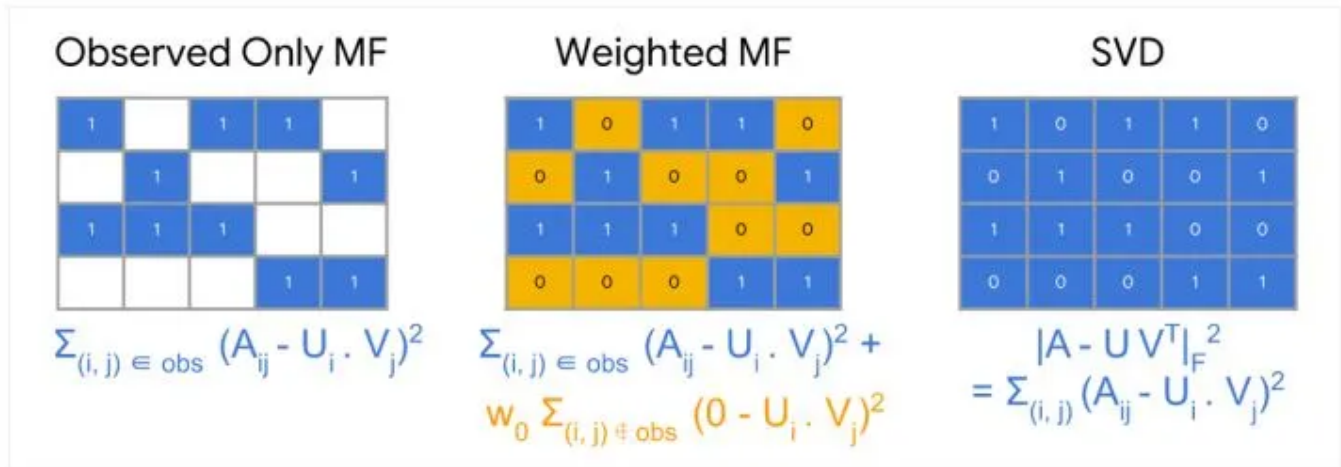
注意：与学习完整矩阵相比，矩阵分解通常会得到更简洁的表示。完整矩阵具有 $O(nm)$ 项，而 Embedding矩阵具有 $O((n + m) \times d)$ 项，其中Embedding维数 d 通常比 m 和 n 小得多。最终，观测矩阵会被映射到低维子空间中，矩阵分解就可以在数据中找到其对应的潜在信息。在前面的示例中， n ， m 和 d 的值都非常小，因此优势可以忽略不计。但是，在现实世界中的推荐系统中，矩阵分解可以比学习整个矩阵要简洁得多。

3.4 目标函数选择

一种直观的目标函数是距离的平方，即在所有观察到的矩阵项上最小化平方误差之和：

$$\min_{U \in \mathbb{R}^{m \times d}, V \in \mathbb{R}^{n \times d}} \sum_{(i,j) \in \text{obs}} (A_{ij} - \langle U_i, V_j \rangle)^2.$$

在此目标函数中，只求观察到的对 (i, j) 的误差和，即对反馈矩阵中的**非零值**求和。但是，只求1的误差总和并不是一个好方法。最小化矩阵中所有为1元素产生的模型，并不能进行很好的推荐，而且泛化性较差。



也许可以将未观察到的值视为零，并对矩阵中的所有条目求损失和。这样其实就是最小化 \mathbf{A} 及其近似值 UV^T 之间Frobenius距离的平方：

$$\min_{U \in \mathbb{R}^{m \times d}, V \in \mathbb{R}^{n \times d}} \|A - UV^T\|_F^2.$$

可以通过矩阵的奇异值分解（SVD）解决此二次问题。但是，SVD并不是一个很好的解决方案，因为在实际应用中，矩阵 \mathbf{A} 可能非常稀疏。例如，将抖音上所有视频与特定用户观看过的视频进行比较。得到的解 UV^T （对应于模型对输入矩阵的近似值）可能接近于零，从而导致泛化性能较差。

相比之下，加权矩阵分解将目标分解为以下两部分的和：

- 观察到的误差和
- 未观察到的误差和（视作0）

$$\min_{U \in \mathbb{R}^{m \times d}, V \in \mathbb{R}^{n \times d}} \sum_{(i,j) \in \text{obs}} (A_{ij} - \langle U_i, V_j \rangle)^2 + w_0 \sum_{(i,j) \notin \text{obs}} (\langle U_i, V_j \rangle)^2$$

w_0 是调节两项加权的超参，以使目标不被其中一项所支配。调整此超参数非常重要。在实际应用中，还需要仔细权衡观察到的数据。例如，频繁的item（如，非常受欢迎的视频）或频繁的query（例如，重度用户）可能会主导目标功能。可以通过对频繁出现的item所对应的训练样本进行加权来纠正此影响。换句话说，您可以将目标函数替换为：

$$\sum_{(i,j) \in \text{obs}} w_{i,j} (A_{i,j} - \langle U_i, V_j \rangle)^2 + w_0 \sum_{i,j \notin \text{obs}} \langle U_i, V_j \rangle^2$$

$w_{i,j}$ 是query i 和item j 对应的频率函数。

3.5 最小化目标函数

最小化目标函数的常用算法包括：

- **随机梯度下降 (SGD)** 是使损失函数最小化的通用方法。
- **加权交替最小二乘 (WALS)** 专用于此特定目标函数。

在两个矩阵 U 和 V 中，每个目标都是二次的。（需要请注意的是，联合问题并不是凸的）WALS的工作方式是随机初始化Embedding，然后在以下条件之间交替进行：

- 固定 U 求解 V 。
- 固定 V 求解 U 。

每步都可以准确地求解（通过线性问题的解决方法）并可以进行分布式计算。该技术可以保证收敛，因为可以确保每一步都可以减少损失。

3.6 SGD和WALS

下面对SGD和WALS优缺点进行比较：

SGD优点：非常灵活：适用于其他损失函数；可以并行化。

SGD缺点：较慢，收敛速度不那么快；难以处理未观察到的项（entries），需要使用负采样或gravity。

WALS优点：可以并行化；收敛速度比SGD更快；更容易处理未观察到的项（entries）。

WALS缺点：仅适用于平方损失；

3.7 CF的优缺点

优点

- **无需领域知识：**不需要相关领域知识，因为Embedding是自动学习的。
- **发掘用户兴趣：**该模型可以帮助用户发现新的兴趣点。系统可能并不知道用户对给定的item的兴趣度，但是模型仍会推荐给他，因为相似的用户有着相同的兴趣点。
- **很好的初始模型：**在某种程度上，该方法仅需要反馈矩阵即可训练矩阵分解模型。而且该方法不需要上下文特征。实际上，该方法可以用作多个召回队列中的一个。

缺点

- **冷启动问题**

模型预测结果是给定的（用户，商品）相应Embedding的点积。因此，如果在训练数据中item从未出现过，则系统也无法计算其Embedding，也无法得到相应的预测结果。此问题通常称为冷启动问题。但是，以下技术可以在某种程度上解决冷启动问题：

(1) 利用WALS进行预测。给定一个在训练集中未出现的item，如果系统与用户有一些交互，则系统可以很容易计算出该item的Embedding，而无需重新训练整个模型。只需求解以下方程式或其加权形式：

$$\min_{v_{i_0} \in \mathbb{R}^d} \|A_{i_0} - Uv_{i_0}\|$$

上述方程对应于WALS中的一个迭代：用户Embedding保持固定，系统求解item的Embedding。对于新用户也可以这样做。

(2) 启发式生成新item的Embedding。如果系统没有相应的交互信息，则系统可以通过对来自同一类别，来自同一上传者（在视频推荐中）的item的Embedding进行平均来近似其Embedding。

- **难以融入query/item的附加特征**

附加特征是query或itemID以外的其他特征，对于电影推荐，附加特征可能包括国家或年龄。融入可用的附加特征可以提高模型的效果。尽管在WALS中融入诸多特征可能并不容易，但是WALS的泛化模型使这成为可能。

四、基于FM模型召回

FM是Steffen Rendle在2010年提出的，FM算法的核心在于特征组合，以此来减少人工参与特征组合作。对于FM，其优势可分以下三点：

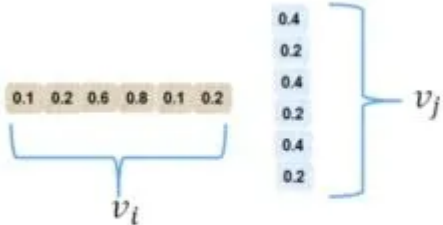
- FM能处理数据高度稀疏场景，SVM则不能；
- FM具有线性的计算复杂度，而SVM依赖于support vector。
- FM能够在任意的实数特征向量中生效。

FM模型

$$\text{FM: } \hat{y}(x) := w_0 + \sum_{i=1}^n w_i x_i + \sum_{i=1}^n \sum_{j=i+1}^n \langle v_i, v_j \rangle x_i x_j$$

LR模型

Dense化两两特征

$$\langle v_i, v_j \rangle := \sum_{f=1}^k v_{i,f} \cdot v_{j,f} =$$




FM的数据结构如下：

	User				Item				Categories				History				Quantity	
x^1	1	0	0	...	1	0	0	...	1	0	1	...	1	0	1	...	2	y^1
x^2	1	0	0	...	0	1	0	...	0	2	1	...	0	0	1	...	4	y^2
x^3	0	1	0	...	1	0	0	...	3	0	14	...	1	0	0	...	5	y^3
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
x^n	0	0	0	...	0	0	1	...	0	1	5	...	0	0	1	...	1	y^n

FM特征数据结构：User相关、Item相关、类别相关的特征、历史行为数据特征等等，最后一列可看作是User对Item评分。FM通过不同特征的组合，生成新的含义。然而，特征组合也随之带来一些问题：

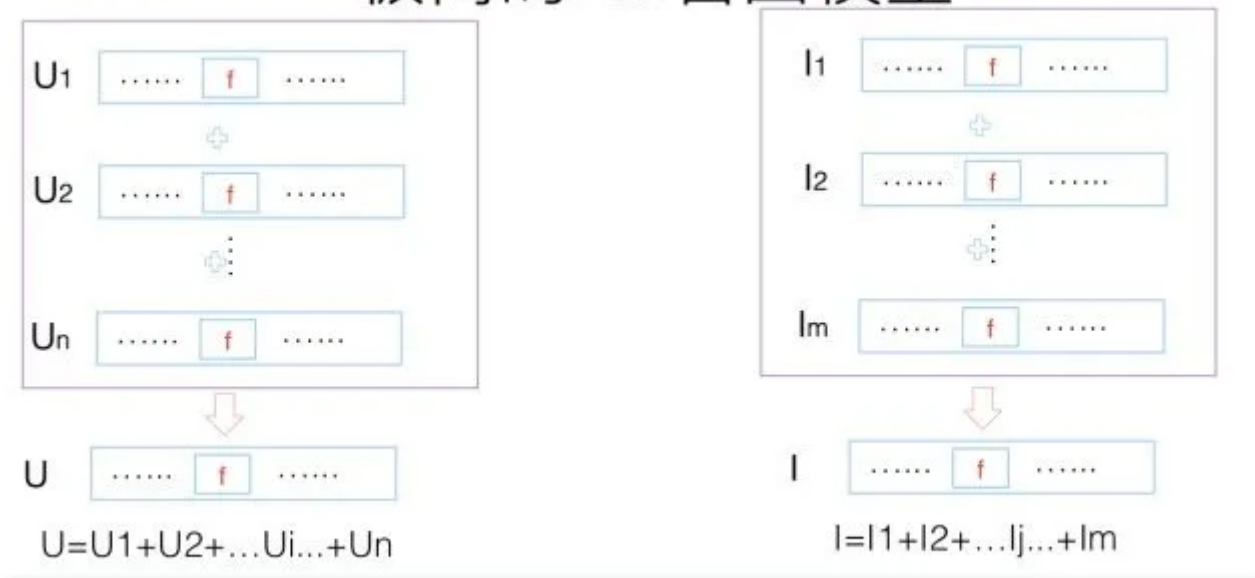
- 特征之间两两组合容易导致维度灾难；
- 组合后的特征未必有效，可能存在特征冗余现象；
- 组合后特征样本非常稀疏，如果原始样本中不存在对应的组合，则无法学习参数，那么该组合就显得无效。

虽然有这些缺点，但是也并不影响FM在广告推荐领域的地位，每个算法都有风靡一时的过去，抱着敬畏之心的态度去学习是没问题的。下面，来看看如何基于FM来做召回的。

4.1 具体召回过程

基于FM的召回与完全版本的FM不同，这里会放弃 U 与 I 特征组内部的二阶交互，即没有了 age、gender、item_id、cate_id 这样的交互，仅是进行到求解隐向量阶段。

极简的FM召回模型



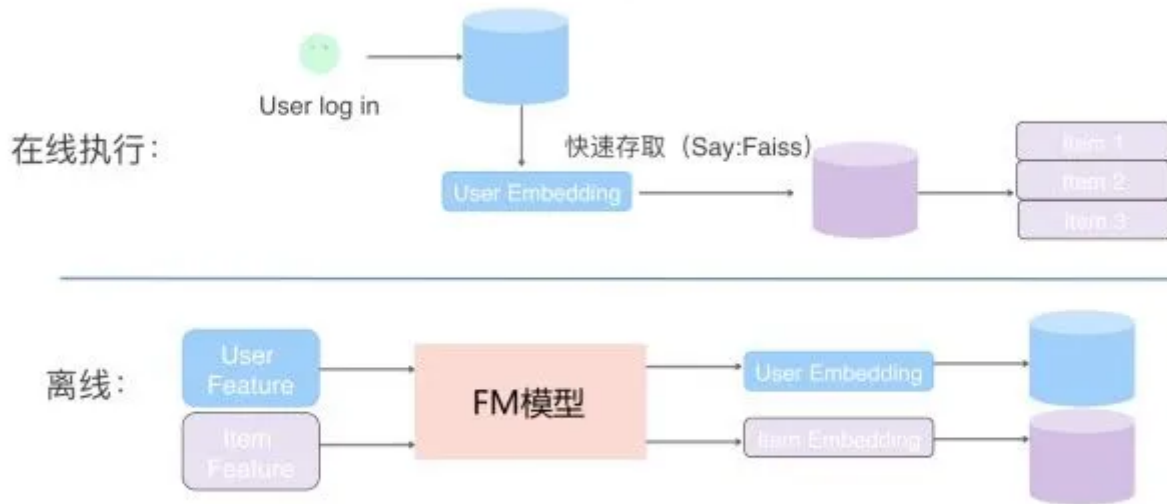
摘自 “推荐系统召回四模型之：全能的FM模型”

这里是 “推荐系统召回四模型之：全能的FM模型” 一文中给出的极简的FM召回模型，即不考虑上下文特征。

第一步，对于某个用户，我们可以把属于这个用户子集合的特征，查询离线训练好的FM模型中这个用户对应的**特征embedding向量**（FM模型求解出的隐向量，即 v_i ，其长度为 k ，包含 k 个描述特征的因子），然后将这个用户对应的n个特征embedding向量累加，形成这个用户的兴趣向量U，这个向量维度和每个特征的维度是相同的。

类似的，我们也可以把每个物品，其对应的物品子集合的特征，查询离线训练好的FM模型对应的特征embedding向量，然后将m个物品子集合的特征embedding向量累加，形成物品向量I，这个向量维度和每个特征的维度也是相同的。

极简的FM召回模型



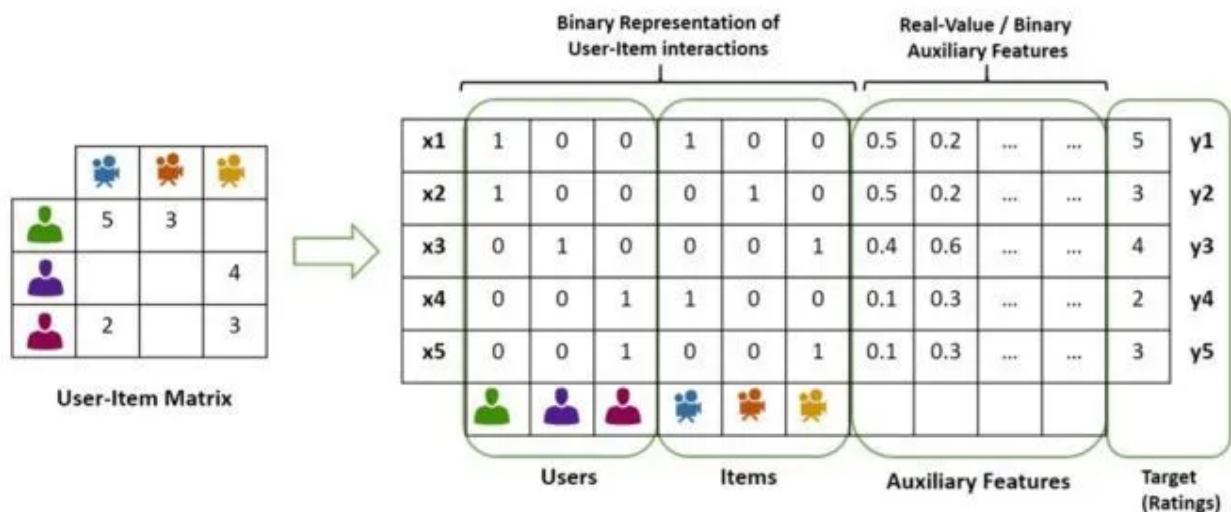
摘自“推荐系统召回四模型之：全能的FM模型”

第二步，对于每个用户以及每个物品，我们可以利用步骤一中的方法，将每个用户的兴趣向量离线算好，存入在线数据库中比如Redis（用户ID及其对应的embedding），把物品的向量逐一离线算好，存入Faiss(Facebook开源的embedding高效匹配库)数据库中，进行knn索引，然后高效检索。

第三步，当用户登陆或者刷新页面时，可以根据用户ID取出其对应的兴趣向量embedding，然后和Faiss中存储的物料embedding做内积计算，按照得分由高到低返回得分Top K的物料作为召回结果。

4.2 矩阵分解和FM

Matrix Factorization到FM的转换



可以认为FM是加了特征的矩阵分解（MF），原来用户和物品侧都只有一个id特征，现在用户侧加了年龄、性别、学历等特征，物品侧加了品类、店铺等特征，然后进一步融入到FM模型后，它将所有的特征转化为embedding低维向量表达，然后用户侧的特征和物品侧特征两两矩阵分解，即两两特征embedding的内积，得到特征组合的权重。

五、基于深度神经网络模型

前文讲述了如何使用矩阵分解来学习Embedding。矩阵分解的一些限制包括：

- 使用附加特征（即queryID /itemID以外的其他特征）困难。因此只能对训练集中存在的用户或item进行推荐。
- 推荐的相关性。正如前文所描述的那样，倾向于向所有人推荐热门item，尤其是在使用点积作为相似性度量时。难以刻画特定的用户兴趣。

深度神经网络（DNN）模型可以解决矩阵分解的这些限制。DNN可以轻松地融入query特征和item特征（由于网络输入层的灵活性），这可以帮助捕获用户的特定兴趣并提高推荐的相关性。

5.1 Softmax DNN模型

一般而言，DNN模型是利用softmax作为最后一层的输出，它会将问题视为多分类问题，其中：

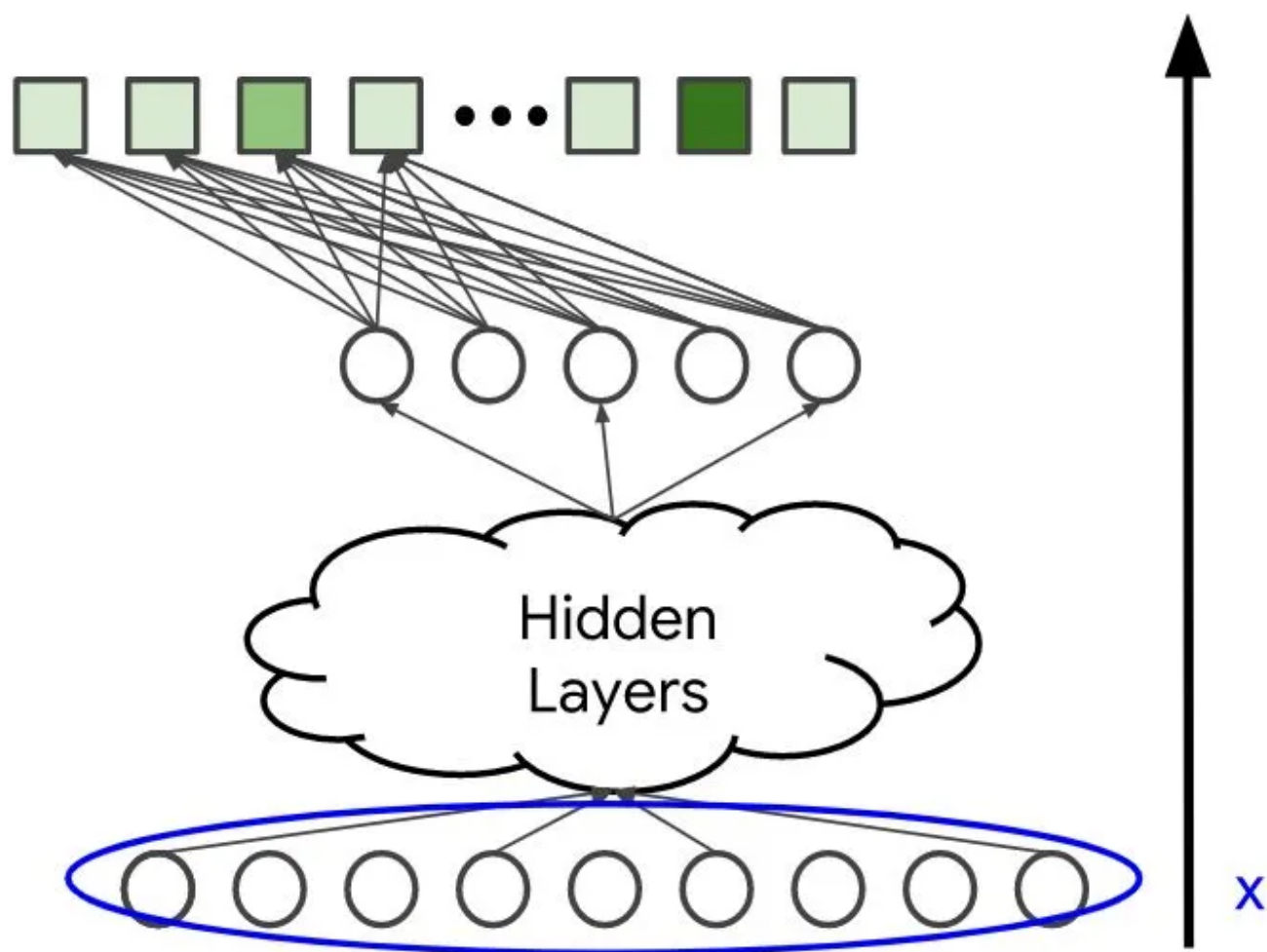
- 输入是用户query。
- 输出是一个概率向量，其大小等于语料库中item的数量，代表与每个item进行交互的概率；例如，点击或观看视频的可能性。

输入层

DNN的输入可以包括：

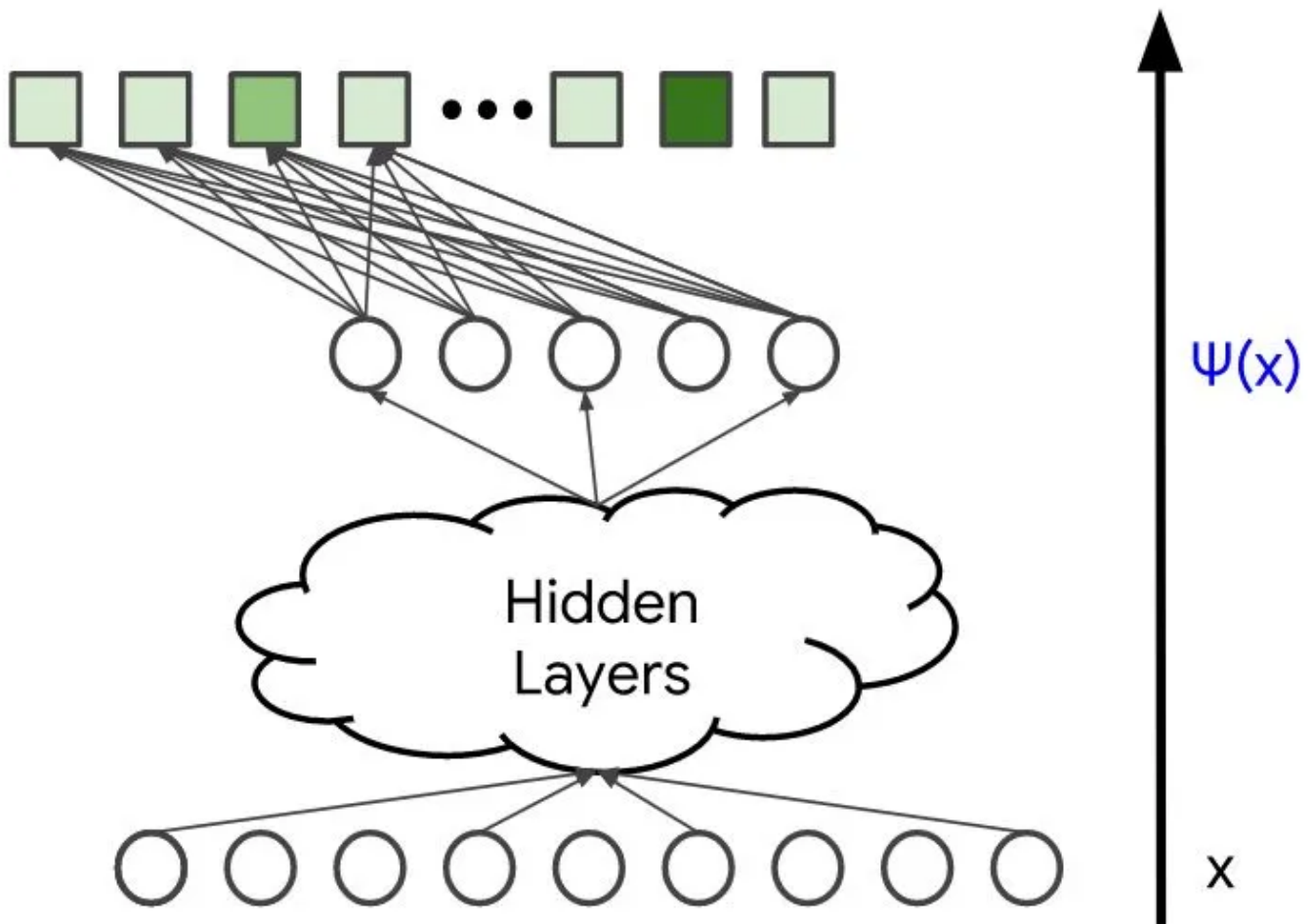
- 稠密(dense)特征（如，点击频率、观看时长等）
- 稀疏(sparse)特征（如，观看视频类型、地区等）

与矩阵分解方法不同，可以添加年龄或地区等附加特征。我们用x表示输入向量。



模型结构

模型结构决定了模型的复杂性和表达性。通过添加隐藏层和非线性激活函数（例如ReLU），模型可以捕获数据中更复杂的关系。但是，增加参数的数量通常也会使模型更难训练且服务成本更高。我们将用 $\psi(x) \in \mathbb{R}^d$ 表示最后一个隐藏层的输出。

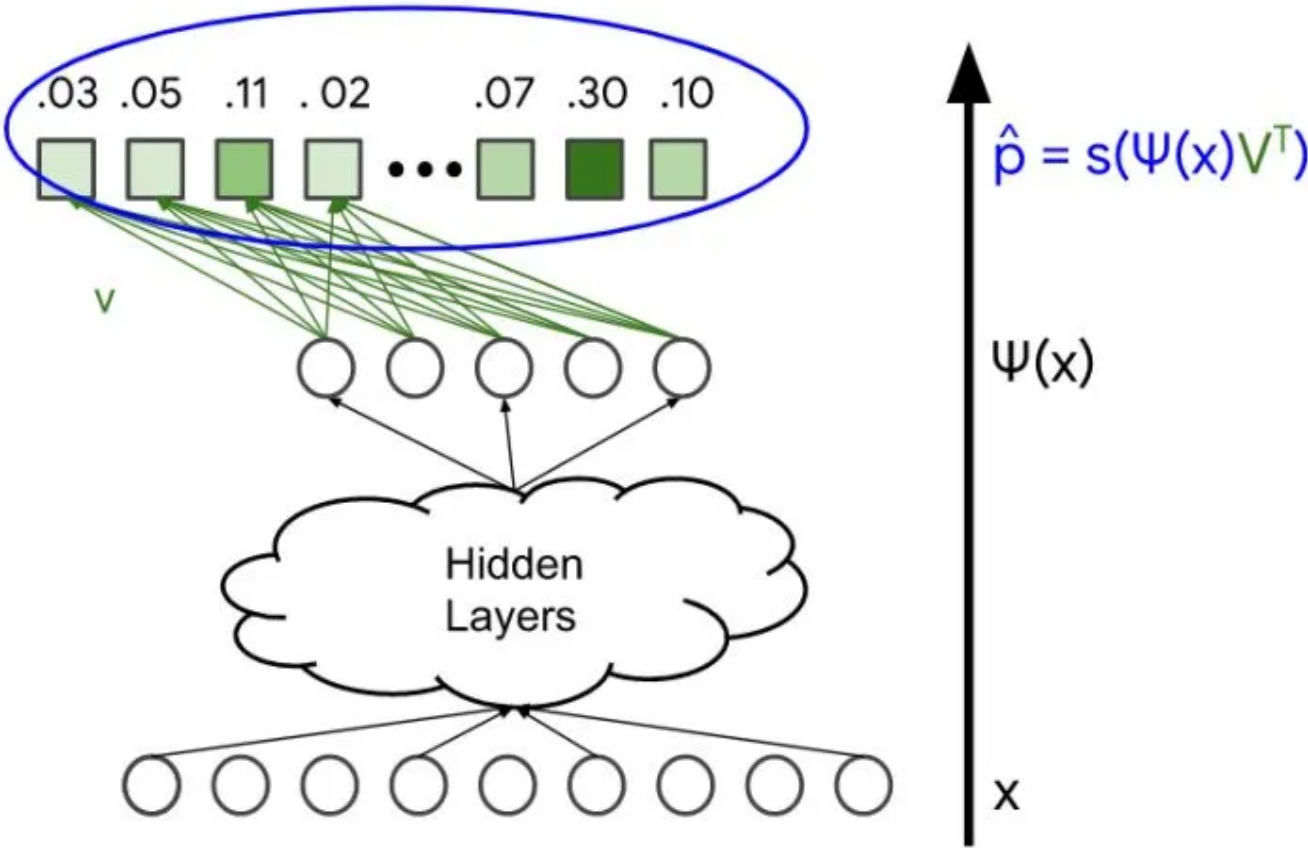


输出层: 预测的概率分布

该模型通过softmax层将最后一层的输出映射到概率分布 $\hat{p} = h(\psi(x)V^T)$ ，其中：

- $h : \mathbb{R}^n \rightarrow \mathbb{R}^n$ 是softmax函数, $h(y)_i = \frac{e^{y_i}}{\sum_j e^{y_j}}$
- $V \in \mathbb{R}^{n \times d}$ 是softmax层的权重矩阵。

softmax层将分数矢量 $y \in \mathbb{R}^n$ （有时称为logits）映射到概率分布。

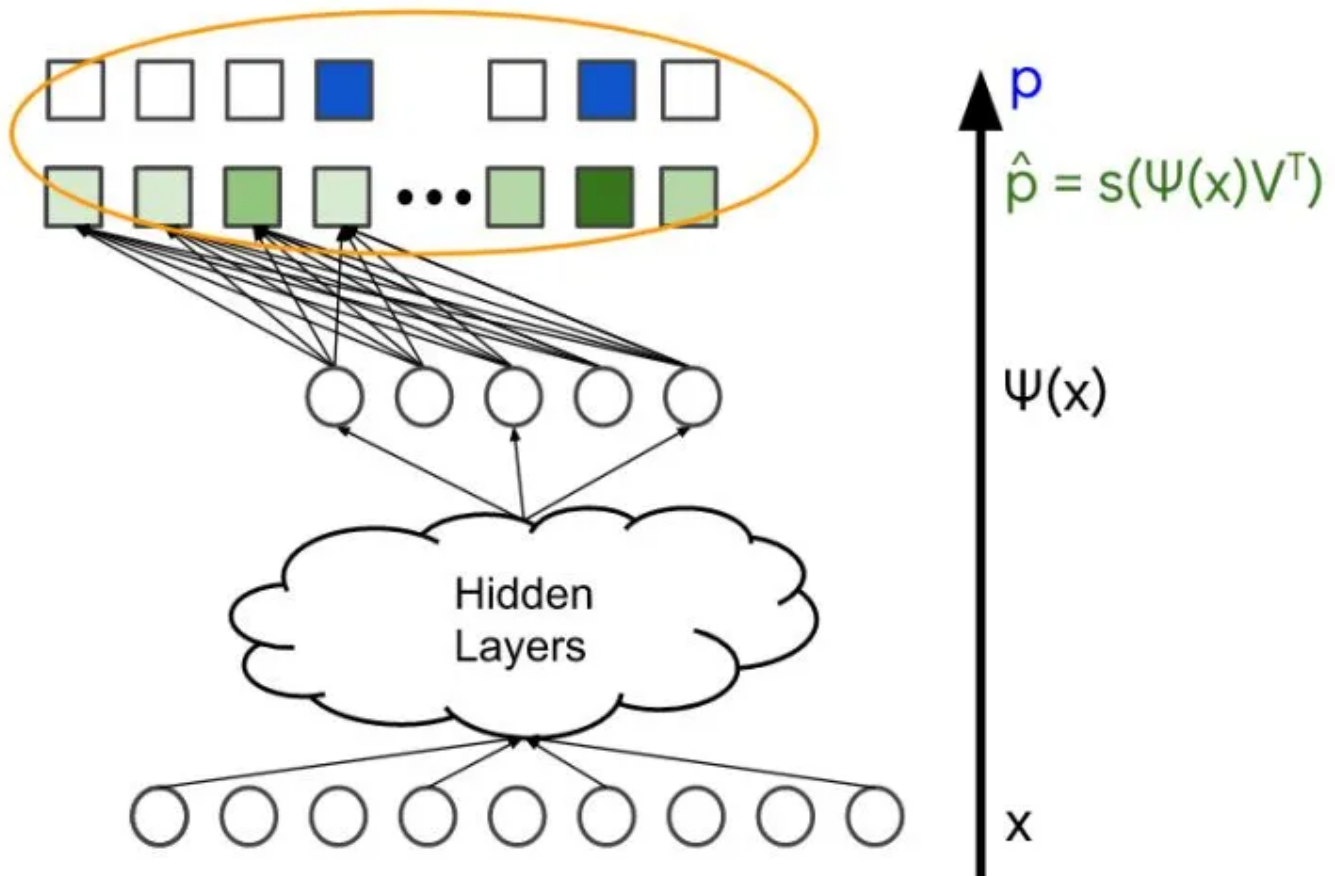


损失函数

最后，定义用以比较以下两项的损失函数：

- \hat{p} , softmax层的输出（概率分布）
- p , ground-truth, 代表用户与之互动的item（例如，用户点击或观看的视频）。这可以表示为归一化的muti-hot分布（概率向量）。

例如，可以使用交叉熵损失来比较两个概率分布。

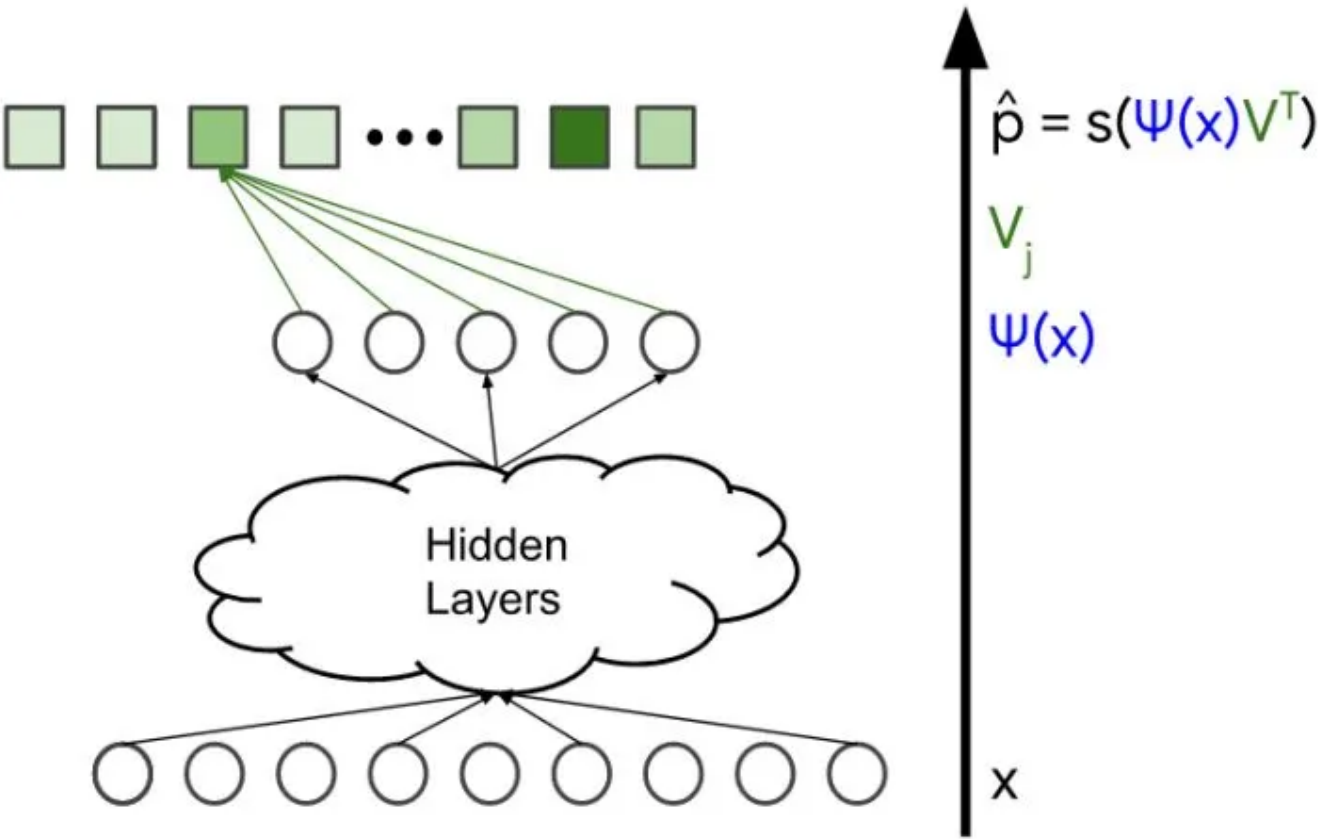


Softmax Embedding

$item_j$ 的概率由 $\hat{p}_j = \frac{\exp(\langle \psi(x), V_j \rangle)}{Z}$ 给出，其中 Z 是不依赖于 j 的归一化常数。

换句话说， $\log(\hat{p}_j) = \langle \psi(x), V_j \rangle - \log(Z)$ ，因此， $item_j$ 的对数概率是（最大为加法常数）两个 d 维矢量的点积，可以将其解释为query和item Embedding：

- $\psi(x) \in \mathbb{R}^D$ 是最后一个隐藏层的输出。我们称其为query 的Embedding。
- $V_j \in \mathbb{R}^D$ 是将最后一个隐藏层连接到输出 j 的权重向量。我们称其为item的Embedding。

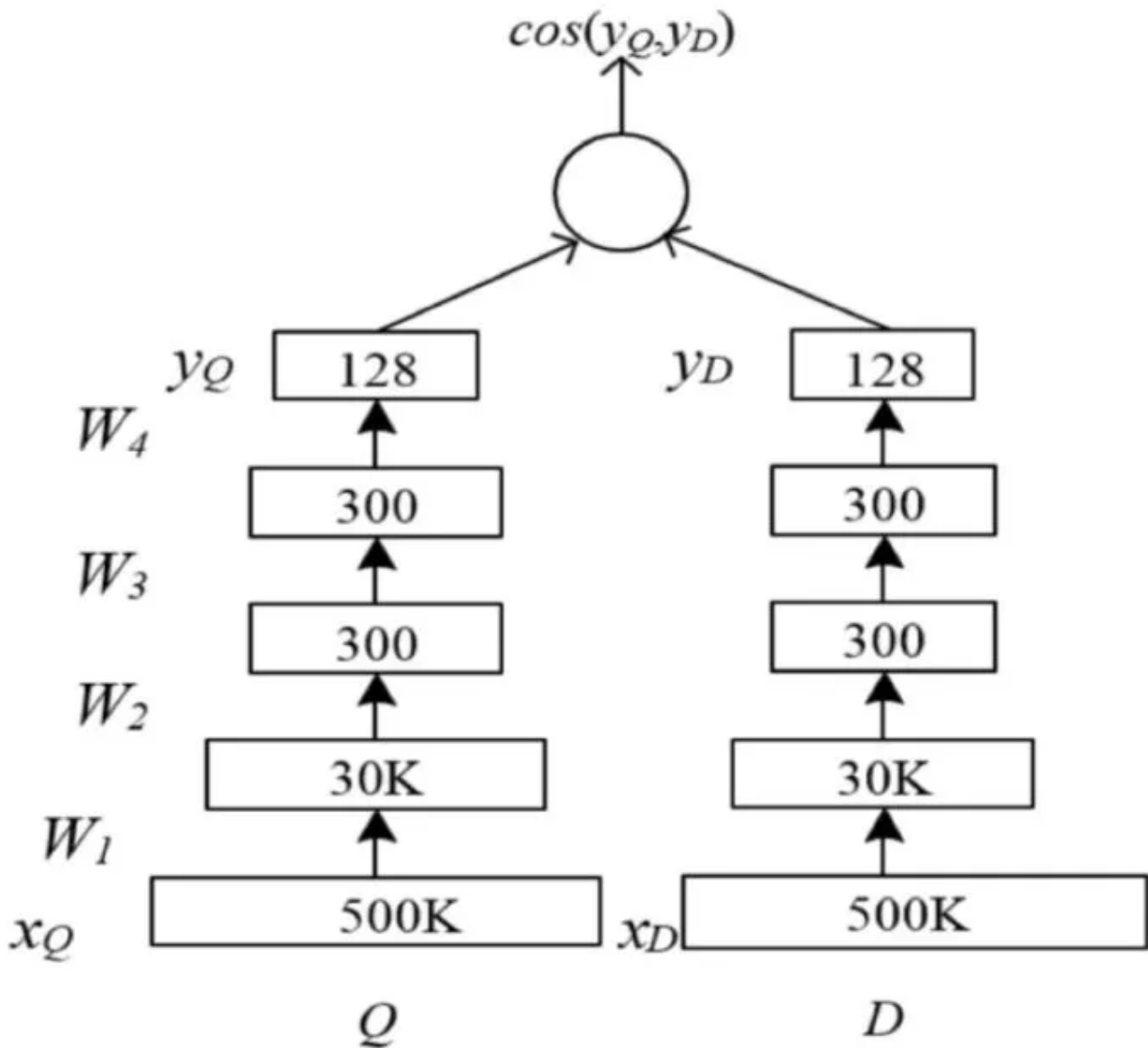


5.2 DNN和矩阵分解

在softmax模型和矩阵分解模型中，系统对于每个item学习一个Embedding向量 V 。我们在矩阵分解中所谓的item Embedding 矩阵 $V \in \mathbb{R}^{n \times d}$ 是softmax层的权重矩阵。但是，用户query Embedding是不同的。将不再对每个query学习一个对应的query Embedding向量，而是学习从query 特征 x 到Embedding的映射 $\psi(x) \in \mathbb{R}^D$ 。因此，可以将此DNN模型视为矩阵分解的泛化，其中将query侧替换为非线性函数 $\psi(\cdot)$ 。

Item特征的使用

可以将相同的想法应用于item侧吗？也就是说，除了对每个item学习一个对应的Embedding之外，模型可以学习将item特征映射到Embedding的非线性函数吗？当然可以。可以使用由两个神经网络组成的双塔模型：



如上图所示，DSSM模型最早被提出是用来提升搜索场景下文档和query匹配的问题，模型接受两个输入到两个神经网络中，通过模型编码到同一个语义向量空间。对于推荐排序场景：

- 一种神经网络将query特征 \mathbf{x}_{query} 映射到query Embedding $\psi(\mathbf{x}_{query}) \in \mathbb{R}^D$
- 一个神经网络将item特征 \mathbf{x}_{item} 映射到item Embedding $\phi(\mathbf{x}_{item}) \in \mathbb{R}^D$

模型的输出可以定义为 $\langle \psi(\mathbf{x}_{query}) \phi(\mathbf{x}_{item}) \rangle$ 的点积。请注意，这再是softmax模型。新模型对每对 $(\mathbf{x}_{query}, \mathbf{x}_{item})$ 预测一个值，而不是每个query的概率向量。

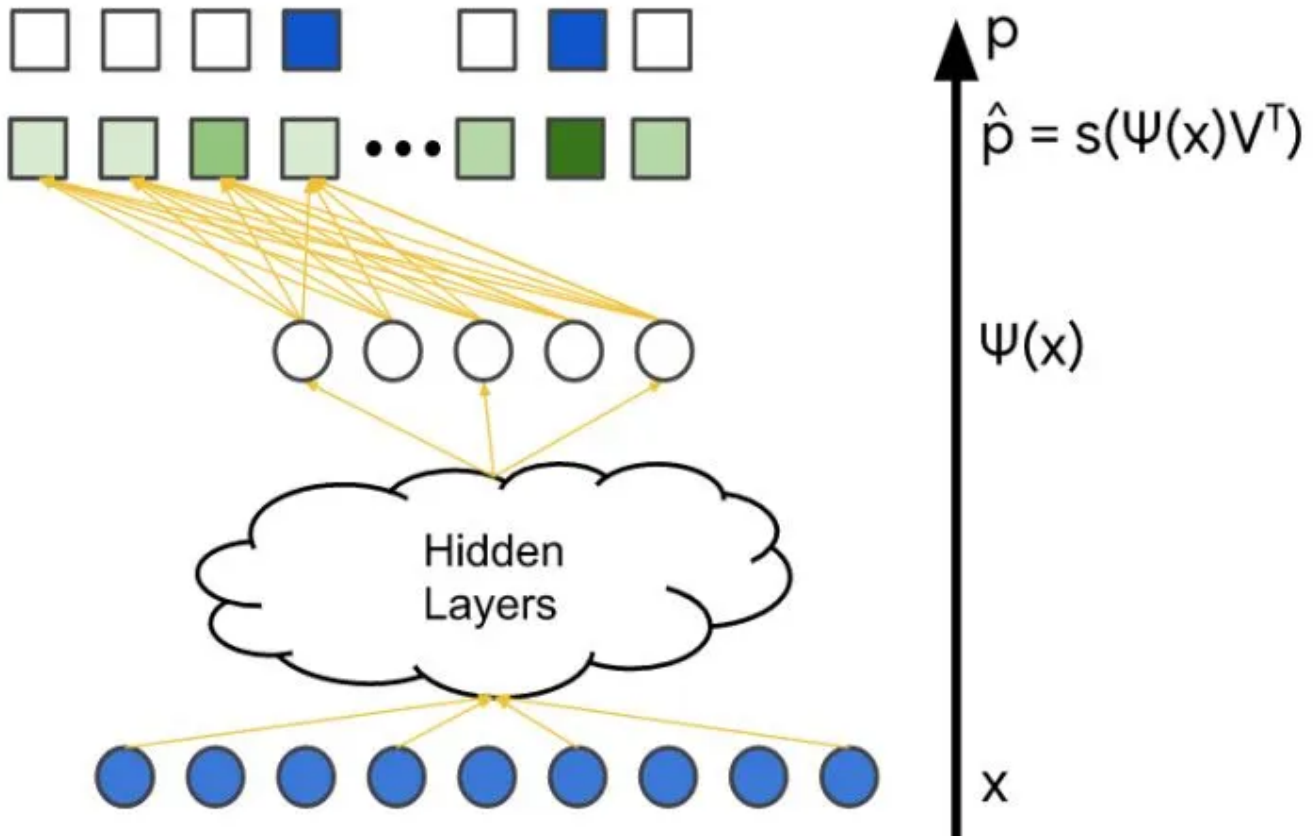
5.3 Softmax训练

前文解释了如何将softmax层合并到推荐系统的深度神经网络中。下面介绍如何利用训练数据对模型参数进行求解。

训练数据

训练数据由query特征 \mathbf{x} 和与用户进行交互的item向量组成（表示为概率分布 \mathbf{p} ）。在下图中，标记为蓝色。模型的变量是不同层中的权重。在下图中，标记为橙色。通常使用随机梯度下降相关方法来

训练模型。



六、更多召回模型

Youtube DNN召回：YouTube在2016年发表的论文《Deep Neural Networks for YouTube Recommendations》为背景进行YouTube的深度神经网络推荐模型的介绍。YouTube的dnn matching召回，将用户和context特征输入DNN，用隐含层最后一层作为向量表示，用Softmax每个item对应的参数作为item的向量表示，通过内积最大索引得到top k

论文地址：Deep Neural Networks for YouTube Recommendations

DSSM语义召回：DSSM模型是微软2013年发表的一个关于query/ doc的相似度计算模型，后来发展成为一种所谓“双塔”的框架广泛应用于广告、推荐等领域的召回和排序问题中。

论文地址：Learning Deep Structured Semantic Models for Web Search using Clickthrough Data

RNN序列召回：基于用户session中的点击序列进行建模召回有很多种方式，其中使用RNN深度网络结构来刻画是其中比较有代表性的一种。相应的网络结构其实很简单，如下图所示。使用用户session中的点击序列作为模型输入，输出则为用户下次点击的item相应的得分。

论文地址：Session-based recommendations with recurrent neural networks

TDM深度树匹配召回：TDM模型是阿里巴巴于2018年提出的新一代深度召回模型，试图通过结合树结构搜索与深度学习模型来解决召回的高性能需求与使用复杂模型进行全局搜索与之间的平衡。它将召回问题

转化为层级化分类问题，借助树的层级检索可以将时间复杂度降到对数级。即认为用户对某节点的兴趣是大于等于其叶子节点的，所以只需在每层选出topk，且在下一层仅计算上一层选出来的节点相应子节点的兴趣，对于规模为M的语料库，只需要遍历 $2 * k * \log M$ 个分支就可以在完全二叉树中找到topk的推荐结果。

论文地址：Learning Tree-based Deep Model for Recommender Systems

参考文献

- <https://zhuanlan.zhihu.com/p/34497989>
- <https://developers.google.cn/machine-learning/recommendation>
- <https://zhuanlan.zhihu.com/p/58160982>
- <https://zhuanlan.zhihu.com/p/87578318>

内容抢先看

在第三讲的文章中，我将会对推荐系统中的排序进行介绍，主要内容分为**建模目标、常见模型、粗排与精排、指标选取**等部分。

点击查看更多竞赛资讯

[深入理解YouTube推荐系统算法](#)

[biendata | U-RISC 识别冠军分享](#)

[视觉竞赛Pipeline（讲义+视频分享）](#)



让我知道你在看