## 论文的组织结构

本文共分为6章，各章节的安排如下：

第一章为绪论，首先介绍了本论文的研究背景及其意义。然后对推荐系统国内外的研究发展进行了梳理，调查了目前推荐算法在程序设计竞赛领域的研究进展。最后简要说明本文的主要工作和组织结构安排。

第二章为相关技术和理论基础，首先对推荐算法的原理及数学推导做了详细的说明，然后对其评测指标做了简要介绍。主要介绍的算法有基于内容的推荐、协同过滤、矩阵分解、Embedding技术和NeuralCF模型。

第三章为系统分析。首先从软件工程的角度对整个系统的功能性需求做出分析，然后对非功能性需求做了约束，最后介绍了系统运行所需的硬件资源。为下面系统详细设计指明了方向。

第四章为系统设计。

第五章为系统实现与结果分析。

# 相关技术与理论基础

## 推荐算法理论研究

### 协同过滤的优化—矩阵分解

随着用户和物品的不断增多，由用户-物品所构成的矩阵会以O(N2)的速度不断增大，用户的历史行为数据也随着被稀释。矩阵分解作为协同过滤算法的优化版本，其算法原理如下图2-3所示：

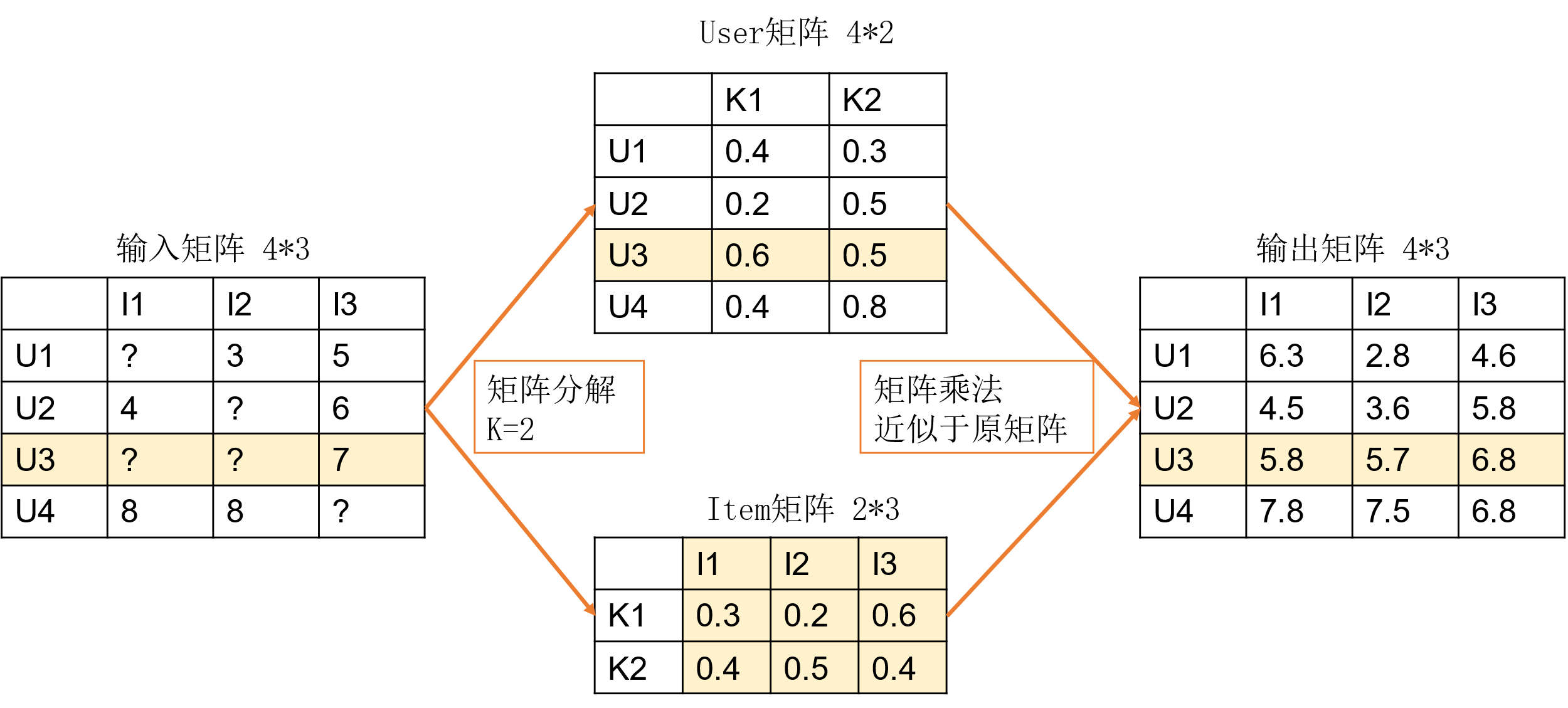


图 2‑3 矩阵分解原理

矩阵分解首先将输入矩阵分解成为为两个子矩阵（User矩阵与Item矩阵）。其中若用户数为M，物品数为N，则User矩阵为维，Item矩阵为维，K表示特征的数量。分解所得到的向量被称之为隐向量，它们用来将用户的兴趣和物品特征关联起来。而输出矩阵通过两个分解矩阵相乘后每个单元格都有值，而这些缺失值的填充值就是对用户还未评价物品的预测值。在相乘之后，输出矩阵不一定和原矩阵一致，而我们可以通过多轮训练，利用梯度下降的方法不断调整K1、K2的参数，最终让这个差值降到最小。

矩阵分解模型最终的输出是在分解后最终产生的用户特征向量和物品特征向量，它们都是K个维度，代表着K个不同的隐含兴趣点。得到用户和物品的隐向量之后，就可以通过相似度计算来进行排序，最终得到我们所需要的推荐列表。

表示用户u对物品i的预估评分，若将其表示两矩阵相乘，则如（式2-7）所示：其中表示物品i在Item矩阵中所对应的列向量，表示用户u在User矩阵中对应的行向量。

为了避免部分噪声点的影响，减少过拟合现状，对（式2-8）加入L2范数作为正则化项后如（式2-9）所示： 为正则化系数，代表约束模型稳定性的能力。

在确定损失函数（式2-9）之后，可以利用梯度下降求解最优化问题。首先，对 求偏导的结果如（式2-10）所示：

对求偏导的结果如（式2-11）所示：

### Embedding在推荐系统中的应用

**（3）如何生成Embedding向量**

生成Embedding向量的方式主要有三种：第一种是基于内容的Word2Vec方法以及后续发展出来的Item2Vec、Node2Vec等词嵌入方式；第二种方法为本文在2.1.4小节中介绍的矩阵分解，其中间生成的隐向量也是一种Embedding向量；最后一种是直接基于DNN深度学习的生成方法。

在介绍Word2Vec算法之前，需要先说明一下在其出现之前人们实现词向量化的方法——One-Hot编码。

One-Hot编码方式为将词典中的每一个词映射为向量中的一个维度，然后将词本身对应的维度置为1，其他维度为0。

这种编码的优点就是简单，只需要将每个词Hash一下就可以为其分配编号，但是随着词数量的增多，One-Hot编码的缺点也显而易见。主要的缺点有三点：

**1. 维度膨胀极快。**随着语料库的增加，编码长度可能突破上亿维。

**2. 两个向量之间的表示没有任何联系。**也就是说我们不能通过One-Hot编码之间的相似度运算得到任何有用的信息。

**3. 数据稀疏。**经过One-Hot编码后的向量几乎全都为0。

Word2Vec的出现克服One-Hot编码上述存在的缺点。其基本思想是将所有词映射为一个固定长度的短向量，这些短向量构成了一个词向量空间，每个词向量对应了该向量空间中的一个点。那么我们就可以通过计算向量在空间中的距离来判断两个向量之间的相似程度。

而词向量的产生是一个轻型神经网络训练过程中的副产物。Word2Vec有两种经典的模型：CBOW和Skip-gram。它们的结构如下图所示：



图 2‑4 CBOW模型与Skip-Gram模型

这两种模型的区别就在于，CBOW模型是通过中心词周围的词来预测中心词；Skip-gram模型是通过中心词来预测周围词。

CBOW简化版的模型（仅输入一个词）如下图所示：

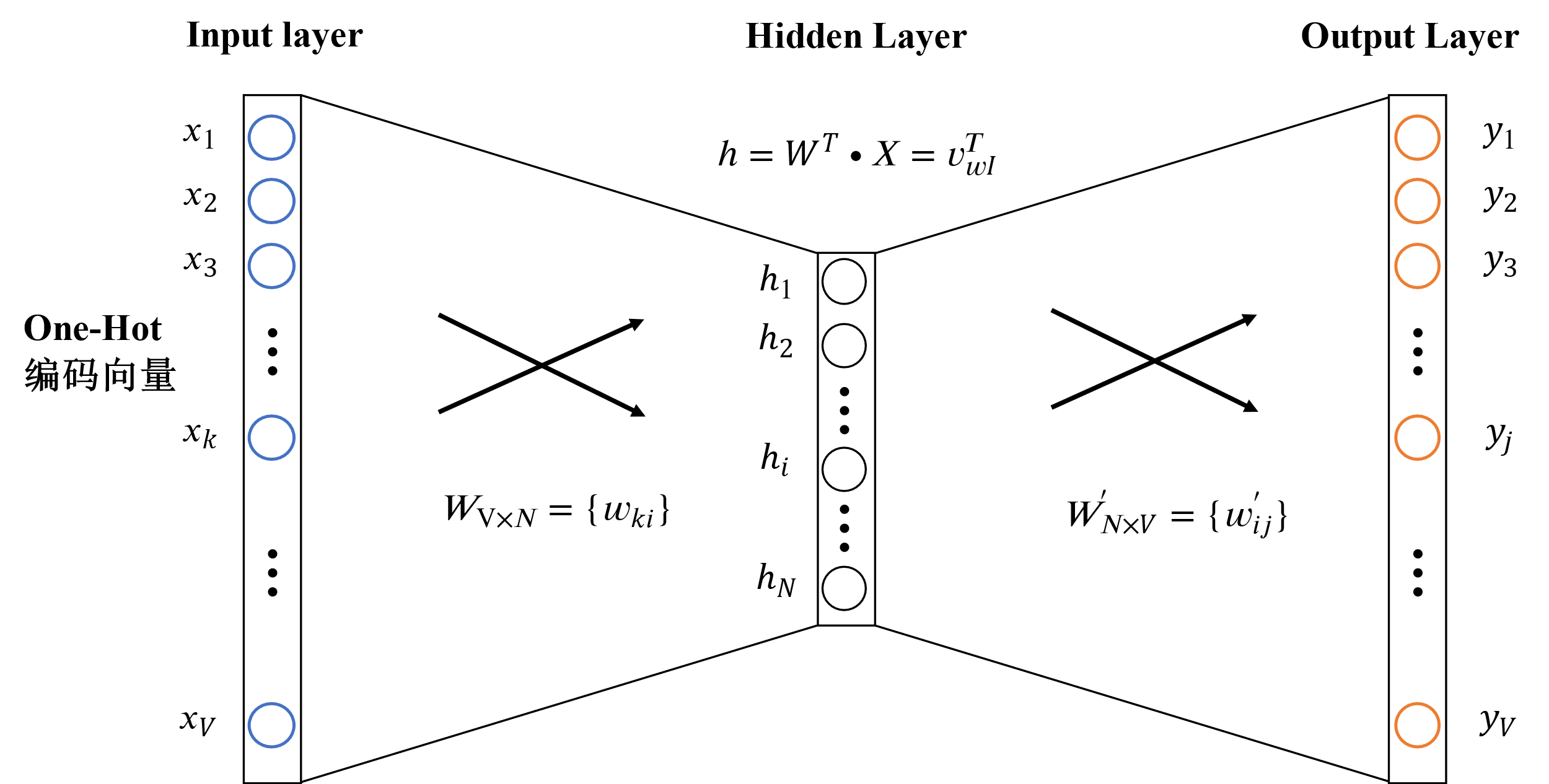


图 2‑5 CBOW简化版模型

**1. 输入层：**约定输入的词库大小为，中间的隐层大小为。若将输入单词标记为，则输入层是其经过One-hot编码后的向量表示，一共有个神经元，除了第个神经元为1，其余各项均为0，例如。

**2. 权重矩阵：**输入层向量到隐层之间的权值可以用一个的矩阵表示，其每一行的值对应着一个我们最终需要的Embedding向量。将One-Hot编码过后的向量映射为一个N维的短向量，在这里我们实现了One-Hot编码的降维。

**3. 隐层：**隐层得到的结果是由输入向量乘权重矩阵得到的。若One-Hot编码的第k位为1，相乘后隐层的值等于权重矩阵的第k行。其数学形式如（式2-14）所示：

**4. 权重矩阵**：在隐层到输出层之间有另一个权重矩阵。通过这个权重矩阵我们可以预估出集合V中每个词可能的得分。其公式如（式2-15）所示：

其中表示权重矩阵的第j列。

**5. 输出层：**在输出时将每一个词的得分通过softmax函数归一化得到一个在之间得概率，其表示为在输入词为时输出词是的概率。可以用（式2-16）对其进行表示：

将（式2-14、2-15）带入（式2-16）整理可得（式2-17）：

其中和分别是输入词的两种不同的N维向量表示形式，其来源于两个不同的权重矩阵的行向量和列向量。

现在定义从输出层到隐层的目标损失函数L为：给定输入词为的权重时，实际输出词为的最大条件概率，是实际输出词的索引。由于使用了softmax函数，所以还需要对其取一个log。目标函数的定义如（式2-18）所示：

确定了目标函数后，就可以利用梯度下降和链式求导规则，对两个权重矩阵进行更新，整个下降的过程如图2-16所示：

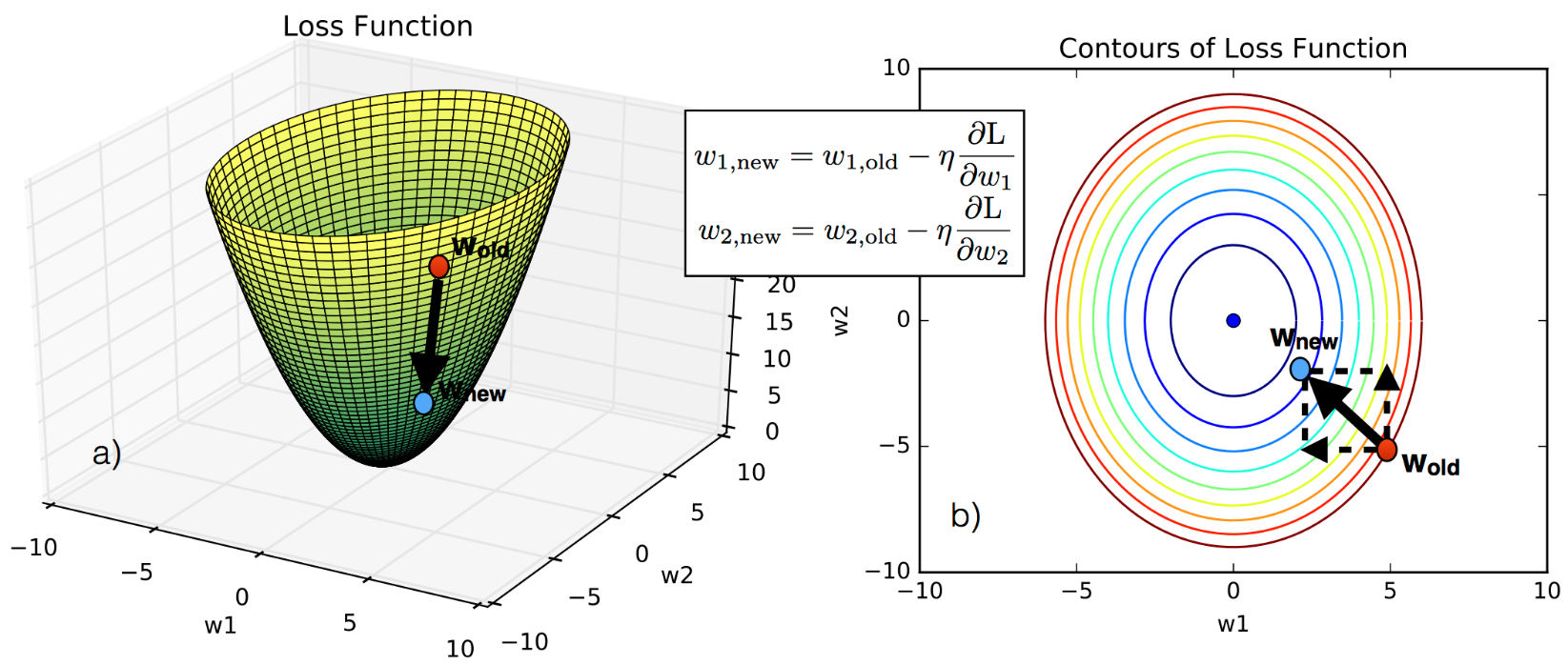
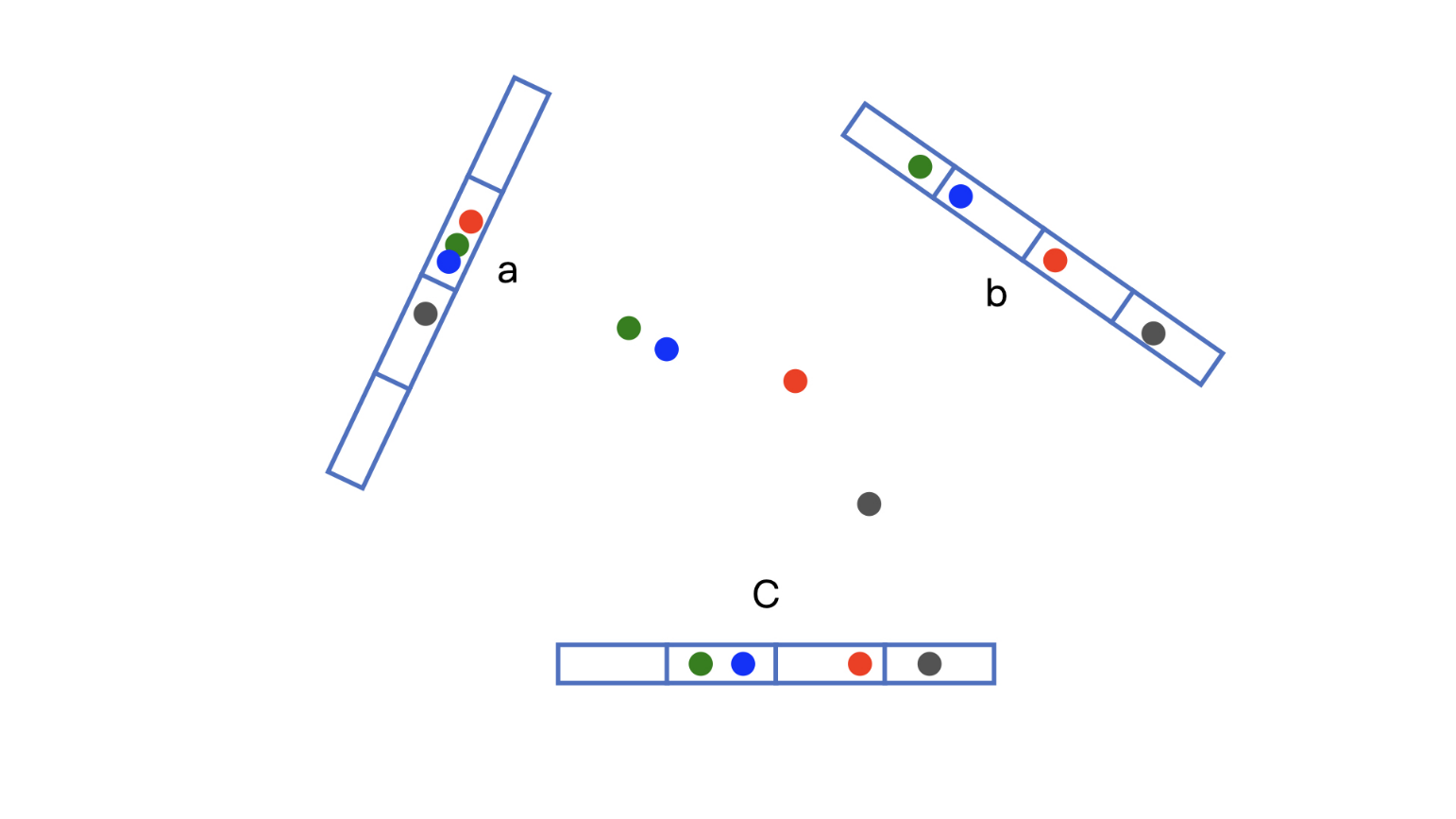


图 2‑6 Word2Vec梯度下降求最优权值

当我们有了Word2Vec之后，若将其（文本ID，文本内容）的输入变为（用户ID，用户提交ID）就可以得到一个行为上的Embedding向量，这就是Item2Vec算法。

**（4）如何快速搜索Embedding向量**

在生成完Embedding向量之后，如何快速得到每个用户最相似的TopN列表有三种方法。第一种方法就是离线使用两层循环暴力为每个用户的Embedding向量挨个计算余弦相似度，然后缓存到Redis里面。这种方法在数据量小的时候还可以使用，一旦数据量过大，离线计算的时间就是我们不能接受的。第二种方法是通过离线局部敏感哈希（LSH）来进行相似搜索。其原理是将Embedding向量分为很多个桶，然后在桶和桶的附近进行搜索，可以极大的增加搜索速度。第三种方法为使用Facebook开源的近邻搜索库Faiss，可以实现ms级别的在线搜索（百万级别）。



上文提到了Embedding向量是数据经过One-Hot编码后映射到高维空间的一个点，而局部敏感哈希就是将其压缩回低维的欧式空间，其原本在高维中相似的点映射到低维依旧靠近。但是原本相隔“距离”较远的点可能会变成接近的点。

假设是高维空间中的维Embedding向量，是随机生成的维映射向量。那利用内积操作可以将映射到一维空间，如（式2-19）所示：

w是分桶宽度，b是0到w间的一个均匀分布随机变量，避免分桶边界固化。

采用 m 个哈希函数同时进行分桶。如果两个点同时掉进了 m 个桶，那它们是相似点的概率将大大增加。

### 深度学习在推荐系统中的应用

NeuralCF

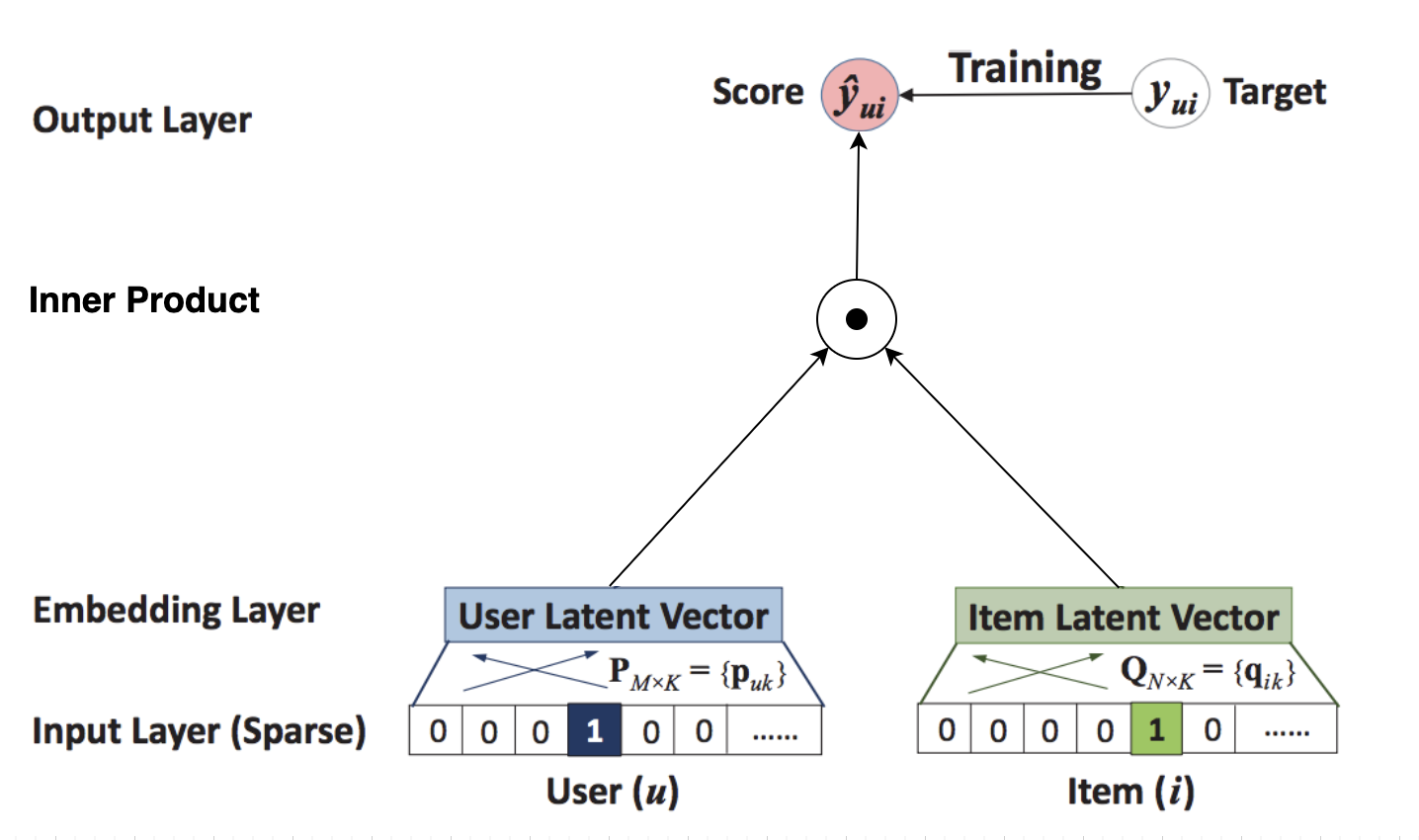


图 2‑7

矩阵分解在 Embedding 层之上直接利用内积得出最终结果。这会导致特征之间还没有得到充分交叉，会有欠拟合的风险。NeuralCF将简单的点积运算变成了多层的神经网络，提高了模型的拟合能力。

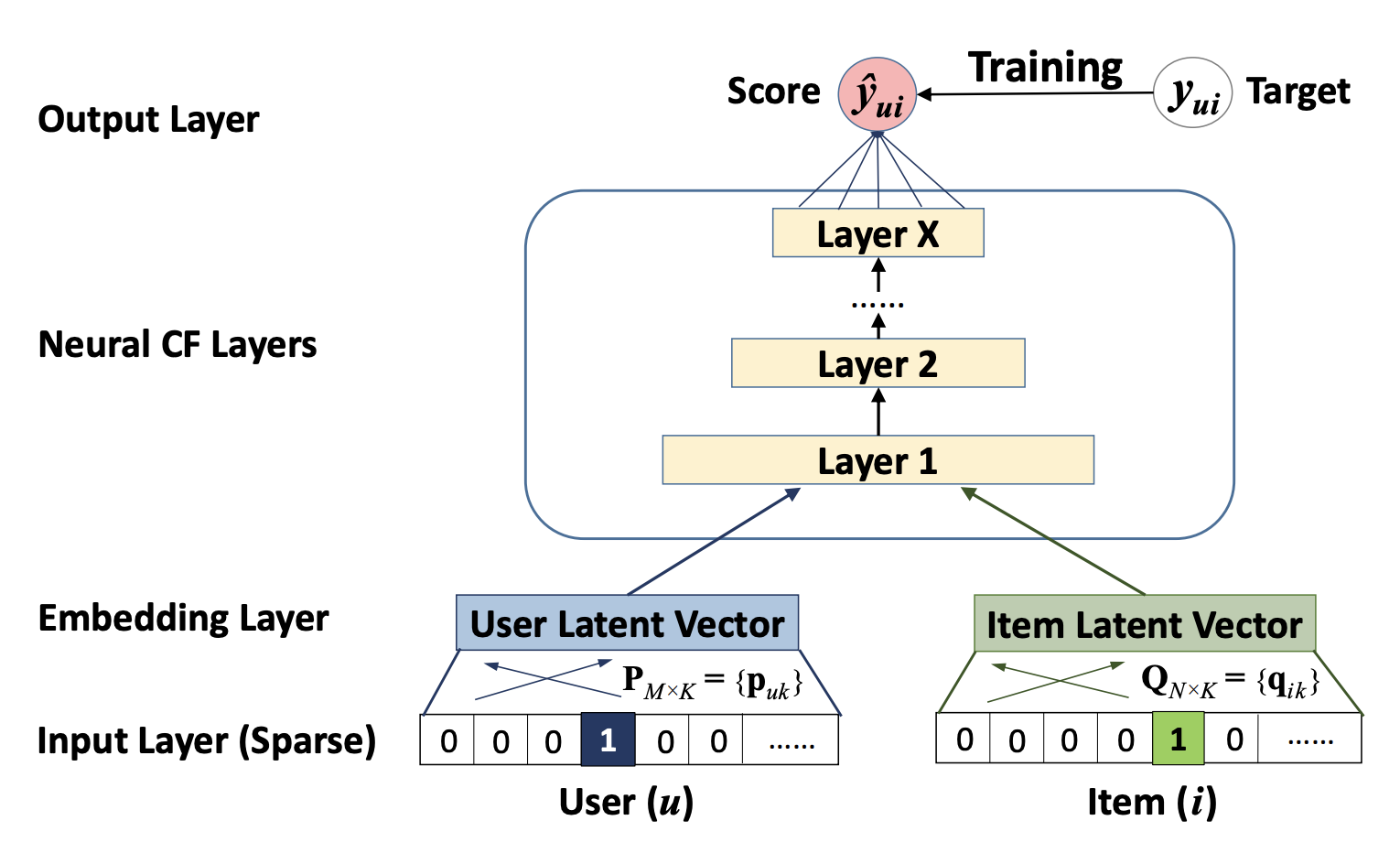


图 2‑8

## 推荐系统评测

### 评测方法

### 评测指标

预测准确度通过比较推荐模型给出的预测评分与用户的真实评分之间

的差距，旨在评估推荐模型与用户的真实兴趣偏好之间的拟合程度。自然，

差距越小，表示推荐算法的预测准确性越好。常用的预测准确度评价公式有平均绝对误差(Mean Absolute Error，MAE)和均方根误差(Root Mean Squared Error，RMSE)。

有时候，知道用户对哪些物品内容感兴趣比能够预测出用户对物品的评分更加重要，因为预测评分高不代表用户一定会选择（观看、购买）该物品。因此，分类准确度一般用在 Top N 推荐的场景中，并使用准确率和召回率这两个指标进行评估

一般情况下，准确率和召回率呈负相关关系，我们自然希望两个指标都能取得比较高的结果。因此，为了综合评估准确率和召回率，还会使用 F1 指标

# 程序设计竞赛组队推荐系统分析

## 系统需求分析

需求分析是在进行系统整体设计之前的一个重要环节，它划清了本系统最终功能实现的范围边界。首先需要对各个模块功能（功能性需求）做出描述，然后对系统在稳定性等非功能性需求上给出约束。

### 功能性需求

功能性需求是指有具体的完成内容的需求，是一个系统必须具备的能力或者遵从的条件。本程序设计竞赛组队推荐系统的功能如下表 3-1 所示：

表 3-1 功能性需求

|  |  |  |
| --- | --- | --- |
| **系统模块** | **功能性需求** | **具体描述** |
| **数据**  **收集**  **模块** | 离线数据收集 | 爬取用户在Codeforces等网站上的历史提交记录，以及整个Codeforces上所有的题目数据信息 |
| 在线日志收集 | 记录用户在系统中的所有操作，如查看文章、提交模板库的修改等操作 |
| **分布式**  **处理模块** | 分布式数据存储 | 将所有收集的来的数据，分布式的存储在三台服务器构成的HDFS系统中，提供给数据处理使用 |
| 分布式数据计算 | 利用分布式的性能优势，进行数据清洗、One-hot编码，Embedding向量生成等操作。 |
| **推荐**  **模块** | 题目个性推荐 | 针对选手在Codeforces网站的历史做题记录，生成符合选手做题风格和难度的推荐列表 |
| 组队队友推荐 | 计算所有选手之间擅长领域的相似程度，选取中等相似区间的未组队选手进行推荐；若组成2人队伍之后以两个人的融合特征做相似计算，为队伍推荐最后一名队员。 |
| 算法新人入门 | 将所有难度系数800分的题目按照做过题目的人数进行排序，生成推荐列表，帮助新人快速熟悉比赛 |
| 算法进阶之路 | 将各个算法类别下的题目按照难度层级进行分层，并在每层下推荐10道做过人数最多的题目。 |
| 近期热门题目 | 将最近一个月时间所有比赛题目中通过和讨论加权次数最多的10道题目生成推荐列表 |
| 历史热门题目 | 将codefoces历史上通过和讨论加权次数最多的10道题目生成推荐列表 |
| **系统业务模块** | 用户注册 | 本推荐系统需要用户提供电子邮箱和登录密码以进行用户注册，完成注册之后方可登录并使用系统。 |
| 用户登录 | 用户需要输入正确的电子邮箱、登录密码以及确认密码，完成系统登录。并使用token保存用户的登录状态使其不必重复登入 |
| 用户登出 | 提供用户登出功能 |
| 忘记密码 | 用户需要通过电子邮箱获取验证邮件，完成密码的找回。 |
| 个人空间 | 将个人的训练数据以柱形图、饼图、雷达图等可视化方法展现 |
| 团队空间 | 提供了训练计划列表，让团队的实力提升有条不紊；提供团队知识覆盖程度可视化，及时发现存在的短板，取得更好的成绩。 |
| 集训空间 | 可视化查看各个队伍的训练情况，如近期通过题目数量、AC题目的平均分数等。 |
| 团队Wiki | 提供一个Wiki平台，允许用户通过提交markdown语法的文本进行Pull Request请求 |

### 非功能性需求

本程序设计组队推荐系统在非功能性需求方面主要有以下几点：

**（1）易用性：**整个系统风格一致、操作简单，不需要专业的数据分析能力即可看出自己的擅长方向和不足之处。但是部分功能还是需要一定的计算机专业素养，如提交Wiki修改时需要懂得如何通过Github提交PR请求。

**（2）通用性：**整个系统应该能够在Windows、Linux、Mac等操作系统上运行。并对访问该系统的电脑无硬件性能要求。

**（3）安全性：**在构建系统时，要能够保证用户的数据不会泄露。具体可包括密码的安全性、服务器抓取的离线数据、训练生成的模型文件等不会遭到数据泄露。

**（4）可靠性与可扩展性：**保证服务的稳定性，不会出现无法访问等严重事故，保证用户的历史行为记录不会丢失。对项目进行合理的划分，将其分为多个模块，并配合详细的项目文档，以便后续版本的更新和加入更多有志于扩展系统的同学进行多人协同开发。

**（5）成熟性和先进性：**在构建整个系统的过程中，使用成熟可靠的软件开发工具，并做到与时俱进，使用业界主流的技术。

## 系统资源说明

本系统实现过程中需要采用的软硬件配置和其功能如下表3-2所示：

表 3-2 硬件配置与功能

|  |  |  |
| --- | --- | --- |
| **名称** | **配置** | **作用** |
| Local | Windows 10, 4C16G, RTX 1050Ti-4G | 前端、后端开发 |
| Master | CentOS 8.0, 2C4G, 无 | Namenode, RM, Master |
| Servant1 | CentOS 8.0, 1C2G, 无 | Datanode, Worker |
| Servant2 | CentOS 8.0, 1C2G, 无 | Datanode, Worker |
| GPU Server | Ubuntu 18.04, 16C62G，RTX 3090-24G | 训练神经网络 |

注：配置项中C代表CPU核数、G代表内存大小，显卡型号后为显存大小。

## 本章小结

本章从软件工程的角度出发，对整个系统进行了需求分析。从功能性需求到非功能性需求，规定了整个系统的功能范围和其他要求，为下面对整个系统进行详细设计指明了方向。