

## 数据预处理

**【内容摘要】** 本章主要讲解数据的预处理。首先简述数据类型、数据特征与数据质量，然后讲述数据采集和抽样的一些方法，最后主要讲述数据预处理的过程，该过程主要包括数据清洗、数据集成、数据变换、数据规约和 Hadoop 中的数据预处理应用。

**【学习目标】** 通过本章的学习，了解数据的类型、特征和质量的概念；掌握常用的数据采集和抽样的方法；熟练掌握应用数据清洗、集成、变换和规约对采集的数据进行预处理，掌握在 Hadoop 中的预处理应用。

数据预处理是实现数据挖掘的首要环节。在当今大数据处理技术中，通过数据预处理过程可产生高质量的有价值的格式规范的易于存储和传输的数据，为进一步进行数据管理、数据挖掘和分析以及可视化展示提供合适的数据源。

### 3.1 数据类型、数据特征与数据质量

实现数据预处理，通常需要基于不同的数据类型，分析数据特征、数据的结构特性等，才能选择合适的处理方法产生有用的数据，保障数据质量。

#### 3.1.1 数据类型

在计算机中，按照不同数据所需内存大小来划分数据类型。数据类型是指程序设计语言中变量所能表示并存储的数据种类。每一种程序语言都提供标准的数据类型，如字符型、整型、浮点型等，同时允许用户自定义数据类型，如结构类型、枚举型、类类型等。

另外，数据类型所占内存的大小也与计算机平台的处理字长相关。在处理字长为 32 位的应用程序中，整型变量所占内存的大小是 4 字节，而在处理字长为 64 位的应用程序中，整型变量所占内存的大小则为 8 字节，但仍然可以设置 4 字节内存长度的短整型变量。

在统计学中，数据类型取决于对统计数据的分类标准或分类方法。

(1) 按照所采用的计量尺度不同，可以将统计数据分为分类数据、顺序数据和数值型数据。

**分类数据：**只能归于某一类别的非数字型数据，属于定性数据或品质数据。分类数据是对事物进行分类的结果，数据表现为类别，是用文字来表达的，它是由分类尺度计量形成的。例如，人口按性别分为男、女。

**顺序数据：**只能归于某一有序类别的非数字型数据，属于定性数据或品质数据。它是由顺序尺度计量形成的，例如将产品分成不同的等级。



数值型数据：按数字尺度测量的观测值，属于定量数据或数量数据。数值型数据是使用自然或度量衡单位对事物进行测量的结果，其结果为具体的数值。

(2) 按照统计数据的收集方法，可以将统计数据分为观测数据和实验数据。

观测数据：通过调查或观测收集到的数据，是在没有对事物人为控制的条件下得到的。

实验数据：在实验中控制实验对象而收集到的数据。

(3) 按照被描述的对象与时间的关系，可将统计数据分为截面数据、时间序列数据和面板数据。

截面数据：在相同或近似相同的时间点上收集的数据，描述的是事物或现象在某一时刻的变化情况。

时间序列数据：在不同时间上收集到的数据，描述的是事物或现象随时间变化的情况。

面板数据：将截面数据与时间序列数据综合起来，按照时间序列和截面两个维度将数据排在一个平面上形成一个表格，称为面板数据(Panel Data)。

在大数据技术中，数据按照大类可分为结构化数据、半结构化数据和非结构化数据。

结构化数据：能够用数据或统一的结构加以表示，称为结构化数据，如数字、符号，又如传统的关系数据模型、行数据等，存储于数据库，可用二维表结构表示。

半结构化数据：所谓半结构化数据，就是介于完全结构化数据(如关系型数据库、面向对象数据库中的数据)和完全无结构的数据(如声音、图像文件等)之间的数据。如 XML、HTML 文档就属于半结构化数据，它一般是自描述的，数据的结构和内容混在一起，没有明显的区分。

非结构化数据：非结构化数据包括所有格式的办公文档、文本、图片、XML、HTML、各类报表、图像和音频/视频信息等，可通过文件系统或非结构化数据库进行存储。

非结构化数据库是指其字段长度可变，并且每个字段的记录又可以由可重复或不可重复的子字段构成的数据库。用它不仅可以处理结构化数据，而且适合处理非结构化数据。

### 3.1.2 数据集与数据特征

数据集(DataSet)又称为资料集、数据集合或资料集合，是一种由数据所组成的集合。规范的数据集是数据存储和管理的需求。针对结构化数据和非结构化数据，数据集通常表现为数据存储使用的表单。

结构化数据集通常以二维表格形式出现。每一列代表一个特定类型的变量，每一行则对应于某一成员的数据集，代表一笔元素。数据集以数据库表单形式存放时，每一列称为一个字段或属性，每一行称为一笔记录或一个元组。最小的数据单位则是元素中的数据项。而对于非结构化数据，按照非关系型数据库如 HBase 的存储方式，数据表的每一行由行键(Row Key)和任意多的列(Column)组成，其中多个列可以组成列簇(Column Family)。每个数据单元(Cell)可以拥有数据的多个版本(Version)，这是使用时间戳来区分的。数据表在行方向上还可划分为多个 Region，Region 是分布式存储的最小单元。在这种非结构化数据存储方式中，数据表示为键/值对(Key/Value)，其中，键由行关键字、列关键字和时间戳构成。

维度、稀疏性和分辨率是大部分数据集所具有的三个基本特性，它们对于数据挖掘技术有不可估量的影响。



维度(Dimensionality)：维度又称维数，是数学中独立参数的数目。在物理学和哲学领域，指独立的时空坐标的数目。数据集的维度是数据集中的对象具有的属性数目。

稀疏性(Sparsity)：稀疏性通常是指在非对称的数据集中，一个对象的大部分属性上的值都为0。在许多情况下，非零项还不到1%。数据的稀疏性不能单一地定义为是一个优点或一个缺点，有时候会节省大量的时间和存储空间，但是会丢失很多数据信息。此外，有些数据挖掘算法仅适合处理稀疏数据。

分辨率(Resolution)：计算机的分辨率可以从显示分辨率与图像分辨率两个方向来分类。描述分辨率的单位有DPI(点每英寸)、PPI(像素每英寸)和LPI(线每英寸)。LPI用于描述光学分辨率。在数据采集过程中，常常可以在不同的分辨率下得到数据，并且在不同的分辨率下数据的性质也不同。多媒体数据或数据模式均依赖于分辨率。

### 3.1.3 探索数据结构

数据结构是数据实体中元素之间的关系，也就是相互之间存在一种或多种特定关系的数据元素的集合。在计算机中，数据结构还包括数据的表示方法和运算，是计算机存储、组织数据的方式。

探索数据结构主要是指探索数据的特征和案例，从而找到数据的独特之处。如样本数据集的数量和质量是否满足模型构建的要求？有没有出现从未设想过数据状态？各因素之间的关系是什么？等等，都是探索数据结构需要考虑的问题。

在机器学习中，收集数据并且把它们加载到R数据结构以后，下一个步骤就是探索数据。对数据理解得深刻，之后的机器学习就可以更加高效。

理解和探索数据的最好方法就是通过实例。例3-1通过R语言针对一些小数据集进行简单数据探索。

**【例3-1】** 对某地区15天的中午空气污染数据进行简单的数据结构的探索。数据的第一行是空气中相关指标的名称，本例主要探索数据的维度和数据中每个指标的数据类型。

```
> kongqi <- read.csv("F:\\kongqi.csv")
> kongqi
   风速 太阳辐射 CO NO NO2 O3 HC
    8      98   7  2   12   8   2
    7     107   4  3    9   5   3
    7     103   4  3    5   6   3
   10      88   5  2    8  15   4
    6      91   4  2    8  10   3
    8      90   5  2   12  12   4
    9      84   7  4   10  15   5
    6      91   4  2   12   7   3
    7      72   7  4   18  10   3
   10      70   4  2   11   7   3
   10      72   4  1    8  10   3
    7      79   7  4    9  25   3
    7      79   5  2    8   6   2
    6      68   6  2   11  14   3
    8      40   4  3    6   5   2
```



```
> head(kongqi)
风速 太阳辐射 CO NO NO2 O3 HC
  8      98  7  2   12   8  2
  7     107  4  3    9   5  3
  7     103  4  3    5   6  3
 10      88  5  2    8  15  4
  6      91  4  2    8  10  3
  8      90  5  2   12  12  4
> str(kongqi)
'data.frame': 15 obs. of 7 variables:
$ 风速 : int  8  7  7  10  6  8  9  6  7  10 ...
$ 太阳辐射 : int  98 107 103  88  91  90  84  91  72  70 ...
$ CO : int  7  4  4  5  4  5  7  4  7  4 ...
$ NO : int  2  3  3  2  2  2  4  2  4  2 ...
$ NO2 : int  12  9  5  8  8  12  10  12  18  11 ...
$ O3 : int  8  5  6  15  10  12  15  7  10  7 ...
$ HC : int  2  3  3  4  3  4  5  3  3  3 ...
```

上面两个运行结果显示,简单的三行代码就显示出导入的数据集的一个结构。`head()`函数展示出此数据集的前六行,`str()`函数提供了一个显示数据框结构的方法,或者说它提供了显示所有同时包含向量和列表的数据结构的方法。语句 15 obs 表示数据一共包含 15 个观测值或者案例。语句 7 variables 表示数据记录了 7 个特征。结合 `head()` 函数展示出来的结果,不难发现,数据一共是 15 个案例和 7 个属性变量。

在变量名后面,int 指出这个属性变量是整型。在这个数据集中,所有的属性变量都是整型的。当然其他的数据还可能出现 chr、num、Factor 等多种数据类型。同时语句 `data.frame` 说明这个数据是数据框型的。

### 3.1.4 数据质量相关概念与数据质量分析

数据质量(Data Quality)是保证数据应用的基础,包括对数据的完整性、一致性、准确性和及时性 4 个方面的要求。

在数据分析和数据仓库方面,数据质量由数据质量元素来描述。数据质量元素(Data Quality Element)是描述数据质量的信息项,包括位置精度、属性精度、逻辑一致性、完整性、现势性和数据说明。数据质量元素分为两类:数据质量的定量元素和数据质量的非定量元素。数据质量定量元素用于描述数据集满足预先设定的质量标准及指标的程度,并提供定量的质量信息。数据质量非定量元素提供综述性的非定量的质量信息。

与数据质量相关的工作包括数据质量管理、控制、评价和审核。这些工作的前提通常需要建立数据质量模型。数据质量模型是用于标识和评定质量信息的形式结构。

数据质量管理(Data Quality Management)是指对数据从计划、获取、存储、共享、维护、应用、消亡生命周期的每个阶段里可能引发的各类数据质量问题进行识别、度量、监控、预警等一系列管理活动,并通过改善和提高组织的管理水平,使数据质量获得进一步提高。

数据质量评价(Data Quality Evaluation)是指对数据质量进行评估的方法和过程。常用的评价方法有演绎推算、内部验证、与原始资料(或更高精度的独立原始资料)对比、独立抽样检查、多边形叠加检查、有效值检查等。在评价过程中,需对每个质量元素进行检查说



明，并给出总的评价，最后形成数据质量评价报告。

数据质量审核(Data Quality Audit)是指对数据质量进行审核分析。在使用新的数据库之前，企业要确认、更正错误数据，并在数据库启用后提供编辑数据的程序。数据质量分析首先进行数据质量审核，即在信息系统中进行数据准确性和完整性方面的结构化调查，它可以在整个数据文件范围内或数据文件范本内调查，也可以调查终端用户对数据质量的看法。

数据质量分析(Data Quality Analysis)是数据挖掘中数据准备过程的重要步骤，是预处理的前提，也是数据挖掘分析结论有效性和准确性的基础。没有可信的数据，数据挖掘构建的模型就是不可靠甚至毫无意义的。

## 3.2 数据采集与抽样

数据采集是实现数据管理和挖掘的先期工作。数据抽样是针对大数据集进行数据分析的必要手段。为了更好地实现数据处理过程，数据的采集和抽样要使用专门的技术手段。

### 3.2.1 数据采集概述

传统的数据采集是指从传感器和其他监测设备等模拟和数字被测单元中自动采集非电量或者电量信号，送到上位机中进行分析处理。

在计算机广泛应用的今天，互联网每天产生海量的数据，这些数据中潜在相当大的应用价值，数据采集就成了重要的环节。获取互联网中的数据依靠人工采集是不现实的，相对高效的方法是应用专门的采集工具，如网络爬虫等来收集有关信息，再经过数据存储、数据预处理、数据挖掘与数据分析，得到有价值的信息。

### 3.2.2 数据采集方法与应用特性

数据采集方法一般包括网上直报、离线填报、Excel 导入、外部数据文件导入、异构数据库导入、主动数据抽取等多种数据采集方式。

网上直报是指用户通过互联网，填报信息，提交到数据库。

离线填报是指将数据通过离线软件采集到计算机中。

Excel 导入是指通过 Excel 表格将数据提交到数据库。

外部数据文件导入是指将计算机中的文件通过特定方法导入数据库，进行有效管理。

异构数据库导入是指通过各种工具将数据在不同数据库类型、不同数据库结构的异构数据库之间进行数据同步。

网络数据采集的主流应用是在网络爬虫。网络爬虫又被称为网页蜘蛛、网络机器人(在 FOAF 社区，经常称为网页追逐者)，是一种按照一定的规则，自动地抓取万维网信息的程序或者脚本。

网络爬虫按照系统结构和实现技术，大致可以分为四种类型：通用网络爬虫(General Purpose Web Crawler)、聚焦网络爬虫(Focused Web Crawler)、增量式网络爬虫(Incremental Web Crawler)和深层网络爬虫(Deep Web Crawler)。实际的网络爬虫系统通



常是几种爬虫技术相结合实现的。

传统爬虫从一个或若干初始网页的 URL 开始,获得初始网页上的 URL,在抓取网页的过程中,不断从当前页面上抽取新的 URL 放入队列,直到满足系统的一定停止条件。聚焦爬虫的工作流程较为复杂,需要根据一定的网页分析算法过滤与主题无关的链接,保留有用的链接并将其放入等待抓取的 URL 队列。接着,它将根据一定的搜索策略从队列中选择下一步要抓取的网页 URL,重复上述过程,直到达到系统的某一条件时停止。另外,所有被爬虫抓取的网页将会被系统存储,进行一定的分析、过滤,并建立索引,以便之后的查询和检索。对聚焦爬虫来说,这一过程所得到的分析结果还可能对以后的抓取过程给出反馈和指导。

网络爬虫可以用 Python、Java、R 等各种语言实现。现今大多数的网络爬虫都用 Python 语言来写。Python 是一门轻量级的面向对象的解释型计算机程序设计语言,它拥有强大的库可以解决各种爬虫问题。

**【例 3-2】**爬取百度贴吧帖子实例,目的是快速获取百度贴吧帖子的标题、楼主和每个楼层的回复信息。

```
# -*- coding:utf-8 -*-
import urllib
import urllib2
import re

class Tool:
    removeImg = re.compile('<img.*?>| {7}|')
    removeAddr = re.compile('<a.*?>|</a>')
    replaceLine = re.compile('<tr>|<div>|</div>|</p>')
    replaceTD= re.compile('<td>')
    replacePara = re.compile('<p.*?>')
    replaceBR = re.compile('<br><br>|<br>')
    removeExtraTag = re.compile('<.*?>')
    def replace(self,x):
        x = re.sub(self.removeImg,"",x)
        x = re.sub(self.removeAddr,"",x)
        x = re.sub(self.replaceLine,"\n",x)
        x = re.sub(self.replaceTD,"\t",x)
        x = re.sub(self.replacePara,"\n      ",x)
        x = re.sub(self.replaceBR,"\n",x)
        x = re.sub(self.removeExtraTag,"",x)
        #strip()将前后多余内容删除
        return x.strip()

class BDTB:
    #初始化,传入基地址,是否只看楼主的参数
    def __init__(self,baseUrl,seeLZ,floorTag):
        self.baseUrl = baseUrl
        self.seeLZ = '? see_lz=' + str(seeLZ)
        self.tool = Tool()
        self.file = None
```

```
self.floor = 1
self.defaultTitle = u"百度贴吧"
self.floorTag = floorTag

def getPage(self,pageNum):
    try:
        url = self.baseURL+ self.seeLZ + '&pn=' + str(pageNum)
        request = urllib2.Request(url)
        response = urllib2.urlopen(request)
        return response.read().decode('utf-8')
    except urllib2.URLError, e:
        if hasattr(e,"reason"):
            print u"连接百度贴吧失败,错误原因",e.reason
            return None

def getTitle(self,page):
    pattern = re.compile('<h3 class="core_title_txt.*?>(.*)</h3>',re.S)
    result = re.search(pattern,page)
    print result.group(1)
    return result.group(1).strip()
else:
    return None

def getPageNum(self,page):
    pattern = re.compile('<li class="l_reply_num.*? </span>.*? <span.*?>(.*)</span>',re.S)
    result = re.search(pattern,page)
    if result:
        return result.group(1).strip()
    else:
        return None

def getContent(self,page):
    pattern = re.compile('<div id="post_content_.*?>(.*)</div>',re.S)
    items = re.findall(pattern,page)
    contents = []
    for item in items:
        content = "\n"+self.tool.replace(item)+"\n"
        contents.append(content.encode('utf-8'))
    return contents

def setFileTitle(self,title):
    if title is not None:
        self.file = open(title + ".txt","w+")
    else:
        self.file = open(self.defaultTitle + ".txt","w+")

def writeData(self,contents):
    for item in contents:
        if self.floorTag == '1':
            floorLine = "\n" + str(self.floor) +
```



```
u"-----\n"
        self.file.write(floorLine)
        self.file.write(item)
        self.floor += 1

def start(self):
    indexPage = self.getPage(1)
    pageNum = self.getPageNum(indexPage)
    title = self.getTitle(indexPage)
    self.setFileTitle(title)
    if pageNum == None:
        print "URL已失效,请重试"
        return
    try:
        print "该帖子共有" + str(pageNum) + "页"
        for i in range(1,int(pageNum)+1):
            print "正在写入第" + str(i) + "页数据"
            page = self.getPage(i)
            contents = self.getContent(page)
            self.writeData(contents)
    except IOError,e:
        print "写入异常,原因" + e.message
    finally:
        print "写入任务完成"
print u"请输入帖子代号"
baseURL = 'http://tieba.baidu.com/p/' + str(raw_input(u'http://tieba.baidu.com/p/'))
seeLZ = raw_input("是否只获取楼主发言,是输入 1,否输入 0\n")
floorTag = raw_input("是否写入楼层信息,是输入 1,否输入 0\n")
bdtb = BDTB(baseURL,seeLZ,floorTag)
bdtb.start()
```

程序结果如下所示。

```
请输入帖子代号
http://tieba.baidu.com/p/xxxxxxxxxx
是否值获取楼主发言,是输入 1,否输入 0
1
是否写入楼层信息,是输入 1,否输入 0
1
该帖子共有 2 页
正在写入第 1 页数据
正在写入第 2 页数据
写入任务完成
Process finished with exit code 0
```

### 3.2.3 数据抽样概述

抽样是一种选择数据对象子集进行分析的常用方法。在统计建模过程中经常会用到抽样技术,通过样本来反映总体的特征。对数据抽样的基本要求是抽取样集的有效性,因此有



效抽样的主要原理就是抽取的样本要与整个数据的效果几乎相同。目前主要有两大类数据抽样技术,即等概率抽样和非等概率抽样。在实际应用中,等概率抽样是常见的。

### 3.2.4 数据抽样方法与应用特性

#### 1. 简单随机抽样

简单随机抽样是现实生活中经常接触到的抽样方法,比如摸彩、抽奖或者抽签决定某个人去做一件事等。简单随机抽样的主要特点就是母群体中的每一个个体都有相同的概率被选入样本。这是一种最公平并且概念上最简单的抽样方法,可以直接用统计学原理去进行估算和推论。

简单随机抽样包括有放回和无放回的抽样,在 R 中使用自带的 sample() 函数就可以实现。sample() 函数的语法为 sample(x, size, replace = FALSE, prob = NULL)。x 表示抽样对象,可以是数值、字符或逻辑向量; size 表示抽样规模,即需要从总体 x 中抽取多少样本; replace 指定是否进行有放回抽样,默认的是无放回抽样,当设置为 TRUE 时,则是有放回抽样; prob 可以指定抽样元素的概率,默认每个个体被等概率抽中。

**【例 3-3】** sample() 函数的简单运用。

```
# 先从 10~100 产生 100 个随机数
>x1 <- runif(100, min = 10, max = 100)
# 接着从上一步中无放回地随机抽取 10 个随机数并输出结果
>sample1 <- sample(x = x1, size = 10, replace = FALSE);sample1
[1] 13.96304 12.26734 69.96037 97.50348 27.99869 54.23512 10.28627 27.14339
[9] 65.59698 35.38356
>x2 <- sample(c('A','B','C','D'), 100, replace = TRUE,
prob = c(0.4,0.3,0.2,0.1))
# 也可以从 A,B,C,D 中随机有放回的抽取 100 个,并且给出每个字母被抽中的概率,然后查看抽到
# 的每个字母的个数
>table(x2);prop.table(table(x2))
x2
A   B   C   D
29 36 27  8
x2
A     B     C     D
0.29 0.36 0.27 0.08
```

#### 2. 系统抽样

系统抽样方法是一种简化的随机抽样法,又叫作等距抽样法。最普遍的做法就是从母群体的数据中,按照一定的间隔抽取足够的个体组成样本。比如一个有 500 个学生的年级,给每个学生编号(1,2,3,4,5,6,7,8,9,10,11,12,...),抽取尾号为“3”的所有同学个体组成样本。

关于系统抽样方法,可以在 R 中使用 sampling 程序包中自带的 UPsystematic() 函数来实现: UPsystematic(pik, eps = 1e-6)。其中,Pik 是一个向量,存放抽样的包含概率;eps 是一个控制值,默认为 1e-6。

**【例 3-4】** UPsystematic() 函数的运用。

```
# 从 1~100 生成 500 个随机数,并保留整数
```



```
>x <- round(runif(500, min = 1, max = 100))
>pik <- inclusionprobabilities(x,100)
>s <- UPSsystematic(pik) #返回 0-1 值表示是否被抽样
>head(getdata(x,s),10)
```

运行结果如下：

```
ID_unit data
      3    74
     11   17
     15   44
     20   88
     25   75
     30   71
     35   96
     40   45
     45   74
     50   99
```

从运行结果中的 ID\_unit 看，并不满足系统抽样的定义，即等间隔地抽取个体组成样本。为了保证与定义的一致性，下面自定义系统抽样的函数。

```
sys_sampling <- function(x, gap = 10, seed = 1234){
  set.seed(seed)
  i <- round(runif(1, min = 1, max = 10))
  ID <- numeric()
  sampling <- numeric()
  while(i<=length(x)){
    ID[ceiling(i/gap)] <- i
    sampling[ceiling(i/gap)] <- x[i]
    i <- i + gap
  }
  return(data.frame(ID = ID, data = sampling))
}
```

上面自定义函数中，x 为待抽样的总体；gap 为抽样间隔，默认为 10；seed 为种子数，用于从 [1,10] 之间随机挑选一个起始号设定随机种子，默认为 123。下面举例验证自定义的函数。

```
>head(sys_sampling(x = x, gap = 7, seed = 3),10)
```

运行结果如下：

```
ID data
 1  3    74
 2 10   30
 3 17   58
 4 24   86
 5 31   27
 6 38   60
 7 45   74
 8 52   76
```