第3章 机器学习初步

本章主要介绍有关机器学习的相关知识，包括机器学习的基本流程、逻辑回归分类、线性回归预测及聚类等。在本章中，将要学习：

- 机器学习概述。
- 机器学习的分类。
- 数据预处理与特征工程。
- sklearn 库简介。
- 逻辑回归分类。
- 线性回归预测。
- 聚类。

3.1 机器学习概述

08 机器学习导引

2016 年 3 月，AlphaGo 与围棋世界冠军李世石进行了围棋人机大战，以 4∶1 的总比分获胜；2017 年 5 月，AlphaGo 与世界排名第一的柯洁对战，以 3∶0 的总比分获胜。围棋界认为，AlphaGo 的棋力已经超过了人类围棋的顶尖水平。随着 AlphaGo 的大火，机器学习（Machine Learing，ML）获得越来越多的关注。

那什么是机器学习呢？美国卡内基·梅隆大学（Carnegie Mellon University）机器学习研究领域的著名教授 Tom Mitchell 对机器学习的定义为：“A program can be said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured bt P, improves with experience E.”翻译过来就是：“如果一个程序在使用既有的经验（E）执行某类任务（T）的过程中被认定为是具备学习能力的，那么它一定需要展现出利用现有的经验（E）不断改善其完成既定任务（T）的性能（P）的特质”。

机器学习是一门多领域交叉学科，涉及概率论、统计学、逼近论、凸分析、算法复杂度理论等多门学科。它专门研究计算机怎样模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构使之不断改善自身的性能。它是人工智能的核心，是使计算机具有智能的根本途径，其应用遍及人工智能的各个领域，它主要使用归纳、综合而不是演绎。

除去一些无关紧要的情况，人们很难直接从原始数据本身获得所需信息，例如，对于垃圾

邮件的检测，检测一个单词是否存在并没有太大的意义，然而当某几个特定单词同时出现时，再辅以考察邮件长度以及其他因素，人们就可以更准确地判定该邮件是否为垃圾邮件。简单地说，机器学习就是把无序的数据转换为有用的信息。

机器学习横跨计算机科学、工程技术和统计学等多个学科，需要多学科的专业知识。它可以作为实际工具应用于从政治到地质学的多个领域，解决其中的很多问题。甚至可以这么说，机器学习对于任何需要解释并操作数据的领域都有所裨益。

机器学习的工作流程如图 3-1 所示。

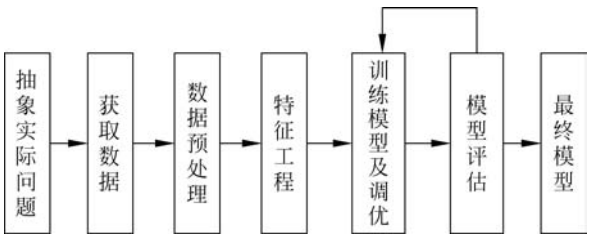


图 3-1　机器学习的工作流程

1. 抽象实际问题

深入理解实际问题的业务场景是机器学习的开始。理解实际问题，主要包括明确可以获得的数据，明确机器学习的目标是分类、回归还是聚类。例如回答一杯液体是红酒还是啤酒，首先要清楚这个问题是一个分类问题，然后就需要从这一杯液体中搜集一些数据，像泡沫数量、液体颜色和酒杯的形状等特征可能是重点，而液体的多少、酒杯的容量等特征可能不需要关注。

2. 获取数据

在获取数据时，得到的数据要有代表性，否则会对结果有很大的影响，会出现过拟合或欠拟合的现象。获取的方式可以是爬虫，可以是数据库拉取，也可以是 API 等。

3. 数据预处理

实际的场景中，得到的数据常常并不满足机器学习算法的要求。因为人为、软件和业务导致的异常数据还是比较多的，例如性别数据的缺失、年龄数据的异常(负数或者超大的数)，而大多数模型对数据都有基本要求，这些异常数据对模型是有影响的。因此，通常都需要对数据进行基本处理，包括数据清洗、数据归一化、扩充等。

4. 特征工程

特征工程包括从原始数据中进行特征构建、特征提取和特征选择。特征工程需要反复理解实际业务场景。特征工程对很多结果有决定性的影响。特征选择好了，非常简单的算法也能得出良好、稳定的结果。特征工程需要运用特征有效性分析的相关技术，如相关系数、卡方检验、平均互信息、条件熵、后验概率、逻辑回归权重等。例如要预测是否下雨，在预测之前，肯定需要一些特征，如是否出现了朝霞或晚霞、温度、空气湿度等。对于分类问题，还需要对数据进行标签，如天气是否下雨等。在这个过程之后，就得到了正式的数据集。一般将数据集分成三组：第一组为用于学习的数据集，称为训练集；第二组用来预防过拟合的发生，辅助训练过程的数据集，称为验证集；第三组用于测试和评估训练好的模型的数据集，称为测试集。为了保证学习有效，需要三个数据集不相交。在实际的运用中，也可以选择训练集和测试集两个数

据集进行学习和测试。

5. 训练模型及调优

根据数据的实际情况和具体要解决的问题来选择模型，要从样本数、特征维度、数据特征等综合考虑，同时，必须清楚解决的问题是分类还是回归。对于模型调优，可以采用交差验证、观察损失曲线、测试结果曲线等分析原因。可以尝试多模型融合，来提高学习效果。

6. 模型评估

根据分类、回归等不同问题，选择不同的评价指标。从各个方面评估模型准确率、误差、时间复杂度、空间复杂度、稳定性、迁移性等，以期达到最佳效果。

过程 5 和 6 可以是一个迭代的过程。当最终模型达到最佳效果后，就可以利用这个模型解决实际问题了。

3.2　机器学习的分类

1. 机器学习的分类概述

机器学习大致分为监督学习、半监督学习和无监督学习，涉及一些相关的技术，如图 3-2 所示。还有一些其他分类，在此不做讨论。

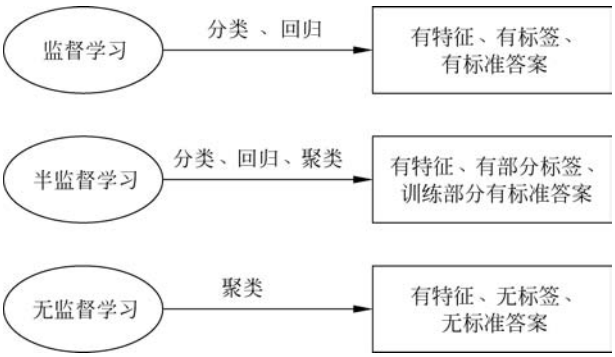


图 3-2　机器学习的分类

监督学习是基于标签训练数据的机器学习模型的过程。假如基于年龄、教育、地点等各种因素去建立一个自动预测人收入的系统。要做到这一点，需要创建一个拥有所有必要细节并标记它的人员的数据库。这样做，是告诉算法，什么参数对应于什么收入。基于这个映射，算法将会学习如何使用提供给它的参数来计算一个人的收入。

半监督学习是模式识别和机器学习领域研究的重点问题，是监督学习与无监督学习相结合的一种学习方法。半监督学习使用大量的未标记数据，以及同时使用标记数据，来进行模式识别工作。当使用半监督学习时，将会要求尽量少的人员来从事工作，同时，又能够带来比较高的准确性，因此，半监督学习目前越来越受到人们的重视。半监督学习不是本章的重点。

非监督学习指的是建立机器学习模型的过程不依赖于标签训练数据。从某种意义上说，它与刚刚讨论的相反。由于没有可用的标签，只能从得到的数据中提取需要的东西。假设想建立一个系统去把一组数据点集分割成多个组。棘手的是不知道分离的标准是什么。因此，一个无监督学习算法需要将给定的数据集以尽可能好的方式进行分组。

在无监督学习中,基本上不知道结果会是什么样子。但可以通过聚类的方式从数据中提取一个特殊的结构。聚类是无监督学习中的一种算法,我们在之后会讨论。在无监督学习中给定的数据和监督学习中给定的数据是不一样的。在无监督学习中给定的数据没有任何标签或者说只有同一种标签,如图 3-3 所示。

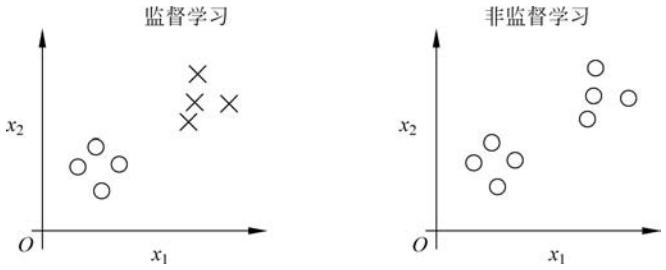


图 3-3　监督学习和非监督学习的区别

2. 分类与回归

在明白了监督学习的思想后,下面介绍监督学习中两类非常重要的应用——分类与回归。

分类的过程是将数据划分为给定的类。在分类过程中,将数据安排到固定数量的类别中以便更有效地使用。通俗来讲,分类就是根据所给数据的属性或者特征是否类似,来把它们归为一类。例如房子,按照房子的级别,可以分为高档住宅、普通住宅、公寓式住宅、别墅等,如图 3-4 所示,这就是分类。

图 3-4　房屋的分类

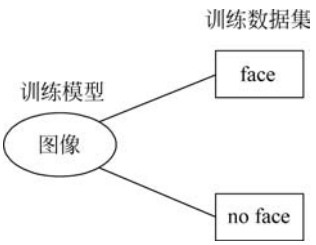


图 3-5　训练数据集

在机器学习中,分类解决了识别新数据点所属类别的问题。我们基于包含数据点和相应标签的训练数据集建立分类模型。例如,假设想要检查给定的图像是否包含一个人的脸。我们将构建一个包含与这两个类相对应的类的训练数据集:face 和 no face,然后根据训练样本来训练模型,如图 3-5 所示。这个经过训练的模型被用于推理。

一个好的分类系统很容易找到数据和检索数据。这在

人脸识别、垃圾邮件识别、推荐引擎等方面得到了广泛的应用。数据分类的算法将会提出正确的标准，将给定的数据分离到给定的类中，如图 3-6 所示。

(a) 人脸识别　(b) 垃圾邮件识别　(c) 推荐引擎

图 3-6　分类系统的应用

机器学习需要提供足够多的样本来推广这些标准。如果样本数量不足，算法就会与训练数据不拟合。这就意味着它在未知数据上不会表现得很好，因为它对模型进行了太多的调整，以适应在训练数据中观察到的模式。这其实是机器学习中经常出现的问题。当构建不同的机器学习模型时这是一个值得考虑的因素。

回归是评估输入变量和输出变量之间关系的过程，如图 3-7 所示。回归从一组数据出发，确定某些变量之间的定量关系式，即建立数学模型并估计未知参数。回归的目的是预测数值型的目标值，它的目标是接受连续数据，寻找最适合数据的方程，并能够对特定值进行预测。这个方程称为回归方程，而求回归方程显然就是求该方程的回归系数，求这些回归系数的过程就是回归，因此，有无数种可能性。

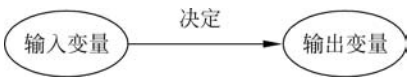


图 3-7　回归

这与分类是相反的。在分类中，输出类的数量是固定的。在回归中，一般认为输出变量取决于输入变量，所以人们想看看它们是如何关联的。输入变量被称为自变量，也称为预测因子，而输出变量被称为因变量，也称为标准变量。输入变量不需要彼此相互独立。在很多情况下输入变量之间存在相关性。

这里简单解释一下分类和回归的区别。分类模型和回归模型本质上是一样的，它们的区别在于输出变量的类型。分类的输出是离散的，回归的输出是连续的。分类问题是从不同类型的数据中学习到数据的边界，例如通过鱼的体长、质量、鱼鳞色泽等维度来分类鲇鱼和鲤鱼，这是一个定性问题；回归问题则是从同一类型的数据中学习到这种数据中不同维度间的规律，去拟合真实规律，例如通过数据学习到面积、房间数、房价几个维度的关系，用于根据面积和房间数预测房价，这是一个定量问题。

回归分析有助于人们理解在保持其他输入变量不变的同时，当改变一些输入变量时，输出变量的值是如何变化的。在线性回归中，假设输入和输出之间的关系是线性的。这限制了建模过程，但它是快速和高效的。在只有一个变量的情况下，线性回归可以用方程 $y=ax+b$ 表示。而如果有多个变量，也就是 n 元线性回归的形式，如：

$$h(x_1,x_2,\cdots,x_n)=\sum_{i=1}^{n}a_ix_i+b$$

有时，线性回归不足以解释输入和输出之间的关系，因此可以使用多项式回归。可以用一个多项式来解释输入和输出之间的关系。这在计算上更复杂，但更精确。回归经常被用于预测价格、经济、变化等。线性回归和多项式回归示例如图 3-8 所示。

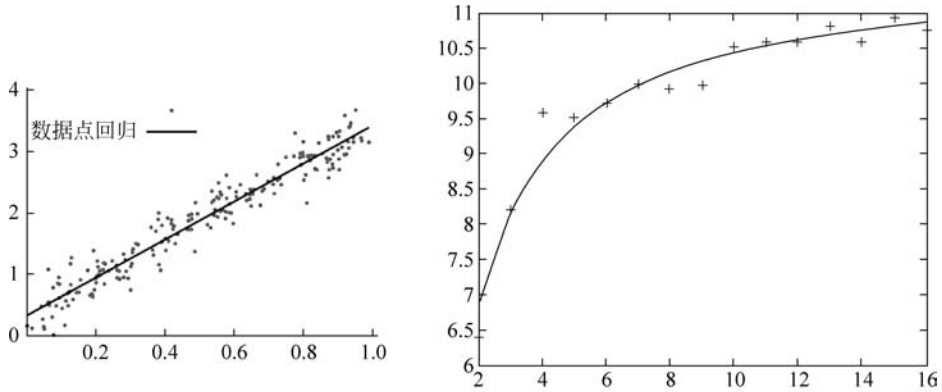


图 3-8　线性回归和多项式回归示例

3.3　数据预处理与特征工程

数据质量对机器学习的算法影响很大，实际业务场景中，大多都需要对数据进行预处理，提高算法的精度。下面介绍几种常用的数据预处理方法。

1. 数据清洗

数据清洗的目的是将数据集中的“脏”数据去除。这些“脏”数据主要包括缺失的数据、异常的数据和重复的数据等。例如，在网上爬取的数据中某个属性可能包括缺失值、个人信息中性别没有填写、人的身高 3m、人的年龄 201 岁等。对于这些“脏”数据，如果量极少，如 10 000 个样本中有 5 个样本是“脏”数据，且是随机出现的，则可以直接删除，因为这 5 个样本对数据集影响不大，但如果有 20%的“脏”数据，直接删除“脏”数据会对整个数据集影响很大。因此要考虑将“脏”数据修改为合理的数据。

对于缺失数据，有以下几种常用处理方法。

(1) 直接删去：这种情况一般限于缺失数据少，删去对数据集影响不大的情况。

(2) 填充为一个常量：例如数值型的数据赋值为 0，文本数据赋值为空或 unknown 等。这样处理效果不一定好，因为算法可能会对这种常量当成数据集本身的属性。

(3) 取均值、中位数或使用频率高的值：选择数据的均值、中位数或使用频率高值进行填充，填充结果可能会存在偏差。

(4) 插值填充：线性插值、拉格朗日插值、牛顿插值。

(5) 模型填充：可以根据数据集的其他属性，通过已知的其他属性值来预测缺失属性值，根据数据类型，定义相应的回归或分类问题。将未丢失数据的那部分样本作为新问题的训练数据。这种方法是最为流行的方法。

异常数据也称为噪声数据。异常数据的发现有以下几种常用处理方法。

(1) 建模法：例如使用回归，找到恰当的回归函数来平滑数据。线性回归要找出适合两个变量的“最佳”直线，使得一个变量能预测另一个变量。多线性回归涉及多个变量，数据要适合一个多维面。那些不能很好拟合的数据，可以判定为异常数据。

(2) 计算机检查和人工检查相结合：可以通过计算机将被判定数据与已知的正常值比

较，将差异程度大于某个阈值的模式输出到一个表中，人工审核后识别出噪声数据。

(3) 聚类：将类似的值组成群或聚类，落在聚类集合之外的值被视为孤立点或离群点，也就是异常数据。孤立点可能是垃圾数据，也可能是提供信息的重要数据。需要根据实际情况进一步处理。

(4) 密度法：如果一个数据的局部密度低于它的大部分临近数据的密度，这个数据可以被认定为是噪声数据。

检测到数据集有噪声数据后，要对数据噪声数据进行处理，处理方式类似于缺失数据的处理方法。

2. 数据变换

数据变换是对对象的属性在数值上进行处理，包括规范化、离散化、稀疏化。下面主要介绍规范化处理。

规范化处理是对数据的归一化和标准化过程。数据中不同的特征由于量纲往往不同，数值间差距可能非常大，会影响到数据分析的结果。需要对数据按照一定比例进行缩放，保持数据所反映的特征信息的同时，使之落在合理范围内，便于进行综合分析。

一般基于样本间距离的机器学习方法，都离不开对数据的规范化处理。有一些模型由于其不关心变量的值，只关心分布情况，可能不需要进行规范化处理。例如基于概率模型的方法、C4.5 分类决策树，依靠数据集关于特征的信息增益比，归一化不会影响结果。

3. 数据过滤

在数据集中，可能某个属性对于整个数据集没有什么意义，影响很小，可以把它过滤掉，例如用户 id 对于判断产品整体购买与未购买数量及趋势就意义不大，直接过滤掉就可以。

4. 特征工程

特征工程是机器学习中最为重要的一部分。什么是特征工程？例如，设计一个身材分类器。输入数据为身高 X 和体重 W，标签为 Y，即身材等级(胖，不胖)。显然，不能单纯地根据体重来判断一个人胖不胖。针对这个问题，一个非常经典的特征工程是 BMI 指数，BMI＝体重/(身高的平方)。这样，通过 BMI 指数就能非常显然地帮助我们刻画一个人身材如何。甚至说可以抛弃原始的体重和身高数据。再例如，数据如果是图像类型，根据学习的目标，要考虑是否获取图像的特征——通道，而不是图像本身。

对于一个真实的数据集而言，可能获取非常多的特征，但是特征并不是越多越好，有的特征可能压根就与实际的结果没有关系，特征数量过多对计算机的开销也会增多。通常来看，会从以下两个大的方面进行选择。

特征是否发散：如果某个特征不发散，特征几乎是不变的，因此也就无法得知该特征对结果的影响。

特征与学习目标的相关性：与学习目标相关性高的特征，肯定是要优先选择的，而与目标几乎不相关的特征可以考虑是否放弃。为了避免特征过多带来学习上的问题，特征降维也被广泛应用。特征降维是特征工程中的一项重要工作，如果数据集的特征很多，不利于算法的学习，需要进行特征降维。特征降维是指从数据集的全部特征选择出一个最优特征子集，在某种评价指标下，训练集和测试集的评估效果最好。


09 sklearn的使用

3.4 sklearn 库简介

经过数据预处理和特征选择过程,得到了机器学习的基本数据集。接下来要根据学习目标,选择相应的学习模型。在没有介绍具体的学习模型之前,介绍一下 Python 中的 Scikit-learn 库的使用,并实现前面介绍的数据预处理和特征工程的基本方法。

Scikit-learn 库由 David Cournapeau 在 2007 年首次开发。它包含一系列容易实现和调整的有用算法,可以用来实现分类和其他机器学习的任务。在官方网站下载时只有 Scikit-learn,但是在 Python 中调用该库时写法为 sklearn,后面在代码中调用该库也均为 sklearn,这里可以将 sklearn 看作是 Scikit-learn 的缩写。

sklearn 的基本功能主要分为六大部分,包括数据预处理、数据降维、模型选择、分类、回归、聚类。sklearn 基本功能如表 3-1 所示。

表 3-1 sklearn 基本功能

基本功能	说 明
数据预处理(preprocessing)	数据特征提取、归一化
数据降维(dimensionality reduction)	主成分分析(PCA)、非负矩阵分解(NMF)、特征选择(eature_selection)等
模型选择(model selection)	pipeline(流水线)、grid_search(网格搜索)、cross_validation(交叉验证)、metrics(度量)、learning_curve(学习曲线)等
分类(classification)	逻辑回归、支持向量机(SVM)、K-近邻、随机森林、逻辑回归、神经网络等
回归(regression)	线性回归、支持向量回归(SVR)、脊回归、弹性回归、贝叶斯回归、Lasso 回归、最小角回归(LARS)等
聚类(clustering)	K-Means(均值聚类)、spectral clustering(谱聚类)、mean-shift(均值漂移)、分层聚类、DBSCAN 聚类

1. 选择数据集

在机器学习过程中,经常需要使用各种各样的数据集,可以找一些通用的数据集来练习使用。在 sklearn 库中提供一些常用的数据集。

(1) 自带的小数据集(packaged dataset): sklearn. datasets. load_< name >,如表 3-2 所示。

表 3-2 自带的小数据集

数据集名称	调用方式	数 据 描 述
鸢尾花数据集	load_iris()	用于分类任务的数据集
手写数字数据集	load_digits()	用于分类任务或者降维任务的数据集
乳腺癌数据集	load-barest-cancer()	简单经典的用于二分类任务的数据集
糖尿病数据集	load-diabetes()	经典的用于回归任务的数据集
波士顿房价数据集	load-boston()	经典的用于回归任务的数据集
体能训练数据集	load-linnerud()	经典的用于多变量回归任务的数据集

（2）可在线下载的数据集（downloaded dataset）：sklearn. datasets. fetch_< name >，如表 3-3 所示。

表 3-3　可在线下载的数据集

数据集名称	调 用 方 式
脸部图片数据集	fetch_olivetti_faces（data_home = None，shuffle = False，random_state = 0，download_if_missing = True）

（3）计算机生成的数据集（generated dataset）：sklearn. datasets. make_< name >，如表 3-4 所示。

表 3-4　计算机生成的数据集

数据集名称	数 据 描 述
make_blobs	多类单标签数据集，为每个类都分配一个或多个正态分布的点集
make_classification	多类单标签数据集，为每个类都分配一个或多个正态分布的点集，提供了为数据添加噪声的方式，包括维度相关性、无效特征以及冗余特征等
make_gaussian-quantiles	将一个单高斯分布的点集划分为两个数量均等的点集，作为两类
make_hastie-10-2	产生一个相似的二元分类数据集，有 10 个维度
make_circle、make_moom	产生二维二元分类数据集来测试某些算法的性能，可以为数据集添加噪声，还可以为二元分类器产生一些球形判决界面的数据

接下来以鸢尾花数据集为例，学习如何在 sklearn 中调用数据集、完成预处理及分类等任务。鸢尾花 iris 分为三个不同的类型：山鸢尾花 Setosa、变色鸢尾花 Versicolor、韦尔吉尼娅鸢尾花 Virginica，分类主要是依据鸢尾花的花萼长度、宽度和花瓣的长度、宽度四个指标。植物学家已经为 150 朵不同的鸢尾花进行了分类鉴定，鉴定的结果放在了这个数据集中。该数据集一般用于监督学习中的多分类问题。我们要解决的问题是：如果自己家的一株鸢尾花开花了，测量了一下花萼的长宽、花瓣的长宽分别是 3.1、2.3、1.2、0.5，然后想知道这朵鸢尾花到底属于哪个分类。

2. 调用数据集

首先要分析这个数据集的组成。在下面的程序中读取数据集并显示基本信息。

程序 3.1　调用数据集

```
1:   from sklearn.datasets import load_iris
2:   iris = load_iris()
3:   print(iris.data)
4:   print(iris.target)              #输出数据所属的真实标签
5:   print(iris.data.shape)          #输出数据的维度
6:   print(iris.target_names)        #输出数据标签的名字
```

输出：

```
[6.9 3.1 5.1 2.3]
[5.8 2.7 5.1 1.9]
[6.8 3.2 5.9 2.3]
```

```
 [6.7 3.3 5.7 2.5]
 [6.7 3.  5.2 2.3]
 [6.3 2.5 5.  1.9]
 [6.5 3.  5.2 2. ]
 [6.2 3.4 5.4 2.3]
 [5.9 3.  5.1 1.8]]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
(150, 4)
['setosa' 'versicolor' 'virginica']
```

分析：输出的内容是部分的数据集内容及基本信息。iris.data 是一个矩阵，有 150 行 4 列数据，每一行数据为花萼长度、宽度和花瓣的长度、宽度四个指标。iris.target 是具体的分类向量，用 0,1,2 代表 3 个不同的类别，类型的名称存储在 iris.target_names 中。

3. 划分数据集

在模型训练时，一般会把数据集划分成训练集、验证集和测试集，其中训练集用来估计模型，验证集用来确定网络结构或控制模型复杂程度的参数，而测试集则用于检验最终选择的最优模型的性能优劣。

sklearn 中使用 sklearn.model_selection 模块对数据集进行划分，而该模块中的 train_test_split()是交叉验证中常用的函数，其功能是从样本中随机按比例选取 train_data 和 test_data，详情如下。

程序 3.2 使用 train_test_split()对数据集进行划分

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
iris = load_iris()
X_train,X_test,Y_train,Y_test = train_test_split(iris.data,iris.target,test_size = 0.4,
random_state = 0)
print('iris 数据集的大小：',iris.data.shape)
print('目标数据集的大小：',iris.target.shape)
print('生成的训练集的特征个数(数据个数)：',X_train.shape)
print('生成的训练集的标签个数(样本个数)：',Y_train.shape)
print('生成的测试集的特征(数据个数)：',X_test.shape)
print('生成的测试集的标签个数(样本个数)：',Y_test.shape)
print('iris 数据集前 5 行的数据：',iris.data[:5])
print('生成的训练集的前 5 行的数据：',X_train[:5])
```

输出：

```
iris数据集的大小： (150, 4)
目标数据集的大小： (150,)
生成的训练集的特征个数（数据个数）： (90, 4)
```

```
生成的训练集的标签个数（样本个数）：  (90,)
生成的测试集的特征（数据个数）：  (60, 4)
生成的测试集的标签个数（样本个数）：  (60,)
iris数据集前5行的数据：  [[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
生成的训练集的前5行的数据：  [[6.  3.4 4.5 1.6]
 [4.8 3.1 1.6 0.2]
 [5.8 2.7 5.1 1.9]
 [5.6 2.7 4.2 1.3]
 [5.6 2.9 3.6 1.3]]
```

4. 数据预处理

归一化：将输入变量变换到某一范围，如[0，1]区间。在 sklearn 库中，使用 MinMaxScaler 类实现；常用于类似梯度下降的优化算法、回归和神经网络中的加权输入以及类似 K-近邻的距离度量。

标准化：通常适用于高斯分布的输入变量。具体来说，将输入变量中的每个属性值减去其平均值，然后除以标准差，得到标准正态分布的属性值。在 sklearn 库中，使用 StandardScaler 类实现；常用于假定输入变量高斯分布的线性回归、逻辑回归和线性判决分析。

正规化：将输入变量变换为具有单位范数长度的数据。常用的范数有 L1、L2。在 sklearn 库中，使用 Normalizer 类实现；常用于含有许多 0 的稀疏数据集，像神经网络采用加权输入的算法和 K-近邻采用距离度量的算法。

二值化：使用门限值，将输入数据变为 0 或 1 两个值。当输入变量值大于门限值时，变换为 1；当输入变量值小于或等于门限值时，变换为 0。在 sklearn 库中，使用 Binarizer 类实现；常用于获取清晰的值的概率，产生新的有意义的属性的特征工程。

程序 3.3　数据预处理练习

```
1:   from sklearn import datasets
2:   import numpy as np
3:   data = datasets.load_iris()
4:   X, y = data.data, data.target
5:   np.set_printoptions(precision = 3)
6:   print ("原始数据: ")
7:   print (X[:4, :])
8:
9:   from sklearn.preprocessing import MinMaxScaler
10:  scaler = MinMaxScaler(feature_range = (0,1))
11:  rescaledX = scaler.fit_transform(X)
12:  # Print transformed data
13:  print ("归一化: ")
14:  print(rescaledX[0:4,:])
15:
16:  from sklearn.preprocessing import StandardScaler
17:  scaler = StandardScaler().fit(X)
```

```
standardizedX = scaler.transform(X)
print ("标准化: ")
print (standardizedX[0:4,:])

from sklearn.preprocessing import Normalizer
scaler = Normalizer().fit(X)
normalizedX = scaler.transform(X)
print ("正规化: ")
print (normalizedX[0:4,:])

from sklearn.preprocessing import Binarizer
binarizer = Binarizer(threshold=0.0).fit(X)
binaryX = binarizer.transform(X)
print ("二值化: ")
print (binaryX[0:4,:])
```

输出:

```
原始数据
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]]
归一化:
[[0.222  0.625  0.068  0.042]
 [0.167  0.417  0.068  0.042]
 [0.111  0.5  0.051  0.042]
 [0.083  0.458  0.085  0.042]]
标准化:
[[-0.901  1.019  -1.34  -1.315]
 [-1.143  -0.132  -1.34  -1.315]
 [-1.385  0.328  -1.397  -1.315]
 [-1.507  0.098  -1.283  -1.315]]
正规化:
[[0.804  0.552  0.221  0.032]
 [0.828  0.507  0.237  0.034]
 [0.805  0.548  0.223  0.034]
 [0.8  0.539  0.261  0.035]]
二值化:
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
```

5. 数据降维

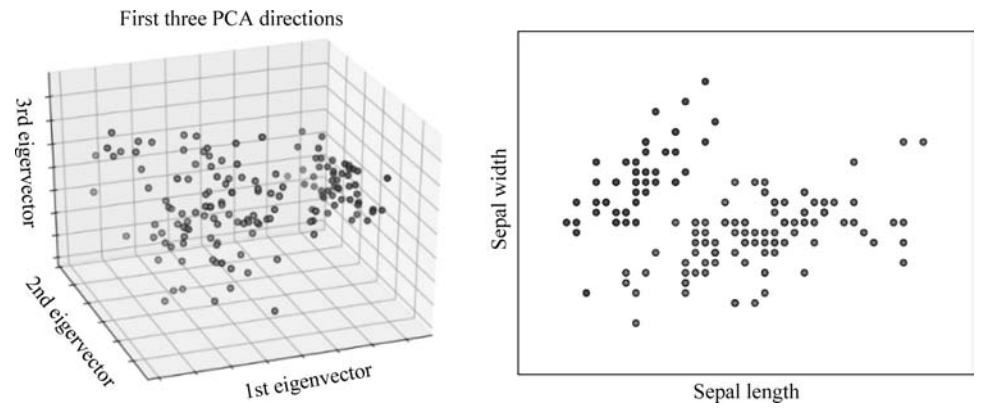
数据降维是指使用主成分分析、非负矩阵分解或特征选择等降维技术来减少要考虑的随机变量个数,其主要应用场景包括可视化处理和效率提升。

程序 3.4　主成分分析

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn import datasets
from sklearn.decomposition import PCA

iris = datasets.load_iris()
X = iris.data[:, :2]          #仅考查前两个特征
y = iris.target
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
plt.figure(2, figsize=(8, 6))
plt.clf()
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1,
                edgecolor='k')
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
fig = plt.figure(1, figsize=(8, 6))
ax = Axes3D(fig,elev=-150, azim=110)
X_reduced = PCA(n_components=3).fit_transform(iris.data)
ax.scatter(X_reduced[:, 0], X_reduced[:, 1], X_reduced[:, 2], c=y,
cmap=plt.cm.Set1, edgecolor='k', s=40)
ax.set_title("First three PCA directions")
ax.set_xlabel("1st eigenvector")
ax.w_xaxis.set_ticklabels([])
ax.set_ylabel("2nd eigenvector")
ax.w_yaxis.set_ticklabels([])
ax.set_zlabel("3rd eigenvector")
ax.w_zaxis.set_ticklabels([])
plt.show()
```

输出：




10 回归

3.5 逻辑回归分类

逻辑回归是用来解释输入变量和输出变量之间关系的一种技术。输入变量是自变量，输出变量是因变量。因变量只能取一组固定的值。这些值对应于分类问题中的类。

逻辑回归的学习过程如图 3-9 所示。

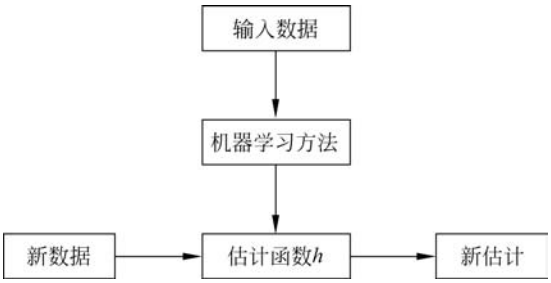


图 3-9 逻辑回归的学习过程

学习的目标是通过使用逻辑函数估计概率来确定自变量和因变量之间的关系。这个逻辑函数是一个 sigmoid()函数，用来构建具有各种参数的函数。它与广义线性模型分析非常接近，试着将一条直线与一堆点相匹配以最小化误差。不用线性回归，而是使用逻辑回归。由于它的简单性使得它在机器学习中很常见。

sigmoid()函数形式为：

$$\text{sigmoid}(x)=g(x)=\frac{1}{1+e^{-x}}$$

逻辑回归虽然名字里带有“回归”，但它实际上是一种分类算法，主要用于二分类问题。逻辑回归通常是利用已知的自变量来预测一个离散型因变量的值(像二进制值 0 和 1)。简单来说，它就是通过拟合一个逻辑函数来预测一个事件发生的概率。所以它预测的是一个概率值，它的输出值应该为 0～1 的一个数值。

sigmoid()函数的分布如图 3-10 所示。

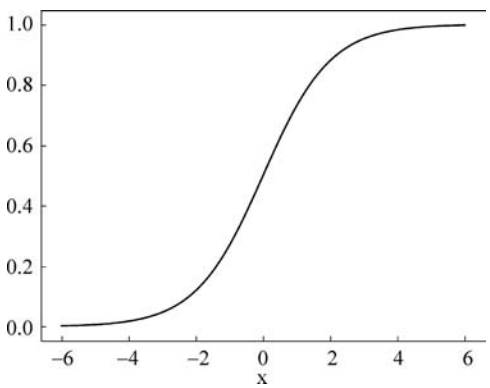


图 3-10 sigmoid()函数的分布

使用 sigmoid()函数，就是让样本点经过运算后得到的结果限制在 0～1，压缩数据的巨幅震荡，从而方便得到样本点的分类标签(分类可以以 sigmoid()函数的计算结果是否大于 0.5

为依据)。

假设你的一个朋友让你回答一个问题。可能的结果只有两种：你答对了或没有答对。为了研究你最擅长的题目领域，你做了各种领域的题目。那么这个研究的结果可能是这样的：如果是一道初中二年级的数学题，你有70%的可能性能解出它。但如果是一道初中一年级的地理题，你会的概率可能只有30%。逻辑回归就是给你这样的概率结果。

逻辑回归主要在流行病学中应用较多，比较常用的情形是探索某疾病的危险因素，根据危险因素预测某疾病发生的概率等。例如，想探讨胃癌发生的危险因素，可以选择两组人群：一组是胃癌组；另一组是非胃癌组。两组人群肯定有不同的体征和生活方式等。这里的因变量就是是否患有胃癌，即“是”或“否”，自变量可以包括很多，例如年龄、性别、饮食习惯、幽门螺杆菌感染等。自变量既可以是连续的，也可以是分类的。

调用 sklearn.linear_model.LogisticRegression()可以实现逻辑回归分类。它使用一些参数，常用的参数如下。

(1) penalty：正则化选择参数。它默认方式为 L2 正则化，可以选用 L1。

(2) C：正则项系数的倒数。

(3) solver：决定了逻辑回归算法中损失函数的优化算法，有四种算法可以选择，分别如下。

① liblinear：使用了开源的 liblinear 库实现，内部使用了坐标轴下降法来迭代优化损失函数。

② lbfgs：拟牛顿法的一种，利用损失函数二阶导数矩阵来迭代优化损失函数。

③ newton-cg：牛顿法的一种，利用损失函数二阶导数矩阵来迭代优化损失函数。

④ sag：随机平均梯度下降，是梯度下降法的变种，和普通梯度下降法的区别是每次迭代仅仅用一部分样本来计算梯度，适合于样本数据多的时候。

lbfgs、newton-cg 和 sag 这三种优化算法时都需要损失函数的一阶或者二阶连续导数，因此不能用于没有连续导数的 L1 正则化，只能用于 L2 正则化。而 liblinear 可以使用 L1 正则化和 L2 正则化。

(4) multi_class：默认值为 ovr，适用于二分类问题。对于多分类问题，用 multinomial，在全局的概率分布上最小化损失。

如果选择了 ovr，损失函数的四种优化方法 liblinear、lbfgs、newton-cg 和 sag 都可以选择。但是如果选择了 multinomial，则只能选择 lbfgs、newton-cg 和 sag。

sklearn 中，所有的估计器都带有 ** fit()和 predict()方法。** fit()用来分析模型，predict()是通过 ** fit()算出的模型，对变量进行预测获得的值。

下面构建一个逻辑回归分类器，用于预测鸢尾花的类别。

程序 3.5　逻辑回归分类

```
1:  from sklearn.datasets import load_iris
2:  import pandas as pd
3:  from sklearn.linear_model import LogisticRegression
4:  import numpy as np
5:  import matplotlib.pyplot as plt
6:  from sklearn.model_selection import train_test_split
7:
8:  iris = load_iris()
```

```
x = iris.data
y = iris.target
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state = 0,test_size = 0.20)
clf = LogisticRegression(C = 1,solver = 'newton - cg',multi_class = 'multinomial')
clf.fit(x_train, y_train)
print("实际值:",y_test)
print("预测值:",clf.predict(x_test))
print(clf.score(x_train,y_train))
print(clf.score(x_test,y_test))
print(clf.predict([[3.1,2.3,1.2,0.5]]))
```

输出：

```
实际值: [2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0]
预测值: [2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0]
0.9666666666666667
1.0
[0]
```

分析：在这个实例中，并没有进行数据预处理等处理，而是直接将数据集分为训练集和测试集，数据集的80%用于训练，20%用于测试。输出的第1行是测试集的实际分类，0、1和2分别表示鸢尾花的三个分类。输出的第2行是学习模型的预测值。对比发现，分类的效果非常好。输出的第3行是模型使用训练集的识别准确率，输出的第4行是模型测试集验证的准确率。第5行是利用模型对[3.1,2.3,1.2,0.5]数据进行分类的结果。

3.6 线性回归预测

线性回归是利用数理统计中回归分析来确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法，运用十分广泛。其表达形式为 $y=wx+e$，e 为误差服从均值为0的正态分布。

回归分析中，只包括一个自变量和一个因变量，且二者的关系可用一条直线近似表示，称为一元线性回归分析。如果回归分析中包括两个或两个以上的自变量，且因变量和自变量之间是线性关系，为多元线性回归分析。

优点：计算比较简单，结果容易理解。

缺点：对非线性数据拟合较差。

下面以sklearn库提供的波士顿房价数据集Boston为例，选用sklearn库中基于最小二乘法的线性回归模型，使用训练集进行拟合，并使用测试集进行验证。

1. 数据集的基本情况

波士顿房价的数据集源于美国某经济学杂志。数据集共有506行14列数据，其中每一行数据都是对波士顿周边或城镇房价的情况描述，每一列数据对应的实际意义如下。

CRIM：城镇人均犯罪率。

ZN：住宅用地所占比例。

INDUS：城镇中非住宅用地所占比例。

CHAS：虚拟变量，用于回归分析。

NOX：环保指数。

RM：每栋住宅的房间数。

AGE：1940 年以前建成的自住单位的比例。

DIS：距离五个波士顿的就业中心的加权距离。

RAD：距离高速公路的便利指数。

TAX：每一万美元的不动产税率。

PTRATIO：城镇中的教师、学生比例。

B：城镇中的黑人比例。

LSTAT：地区中有多少房东属于低收入人群。

MEDV：自住房屋房价中位数(也就是均价)。

sklearn 库提供了不少回归算法，本例利用线性回归算法运行预测，其他的方法可以作为练习。sklearn 提供的常用回归模型如表 3-5 所示。

表 3-5　sklearn 提供的常用回归模型

模块名称	函数名	算法名
linear_model	LinearRegression	线性回归
svm	SVR	支持向量机
neighbors	KNeighborsRegressor	最近邻回归
tree	DecisionTreeRegression	回归决策树
ensemble	RandomForestRegressor	随机森林回归
ensemble	GrandientBoostingRegressor	梯度提升树回归

2. 分析数据集

导入数据后，数据特征很多，一般要做特征选择。在波士顿房价预测实例中，找到与房价最强相关的三个属性。sklearn 库中的 SelectKBest 模块功能是特征选择，可以设置两个参数。

(1) score_func：需要一个得分函数，对于回归问题可以选择 f_regressioin 和 mutual_info_regression；对于分类问题，可以选择 chi2、f_classif 和 mutual_info_classif。默认函数为 f_classif。

(2) k：整数、默认或 all。使用 k 代表选择 k 个特征，默认为 10 个特征。all 选项则绕过选择，用于参数搜索。

SelectKBest 模块提供的常用方法如下。

(1) fit(X,y)：在(X,y)上运行记分函数并得到适当的特征。

(2) fit_transform(X[, y])：拟合数据，然后转换数据。

(3) get_params([deep])：获得此估计器的参数。

下面使用 SelectKBest 模块进行数据集的特征选择。

程序 3.6　波士顿数据集相关性分析

```
1:    from sklearn import datasets
2:    from sklearn.feature_selection import SelectKBest
3:    from sklearn.feature_selection import f_regression
```

```
4:
5:   dataset = datasets.load_boston()
6:   x = dataset.data
7:   y = dataset.target
8:   names = dataset.feature_names
9:   s = SelectKBest(f_regression, k = 3)
10:  s.fit_transform(x,y)
11:  arr = s.get_support()
12:  i = 0
13:  for t in arr:
14:          if t:
15:              print(names[i])
16:  i = i + 1
```

输出：

```
RM
PTRATIO
LSTAT
```

分析：结果输出是 RM、PTRATIO 和 LSTAT 三个特征，与房价相关最高。RM 是每栋住宅的房间数，PTRATIO 是城镇中的教师、学生比例，LSTAT 是地区中有多少房东属于低收入人群。这三个按照实际的意义，还具备一定逻辑性。

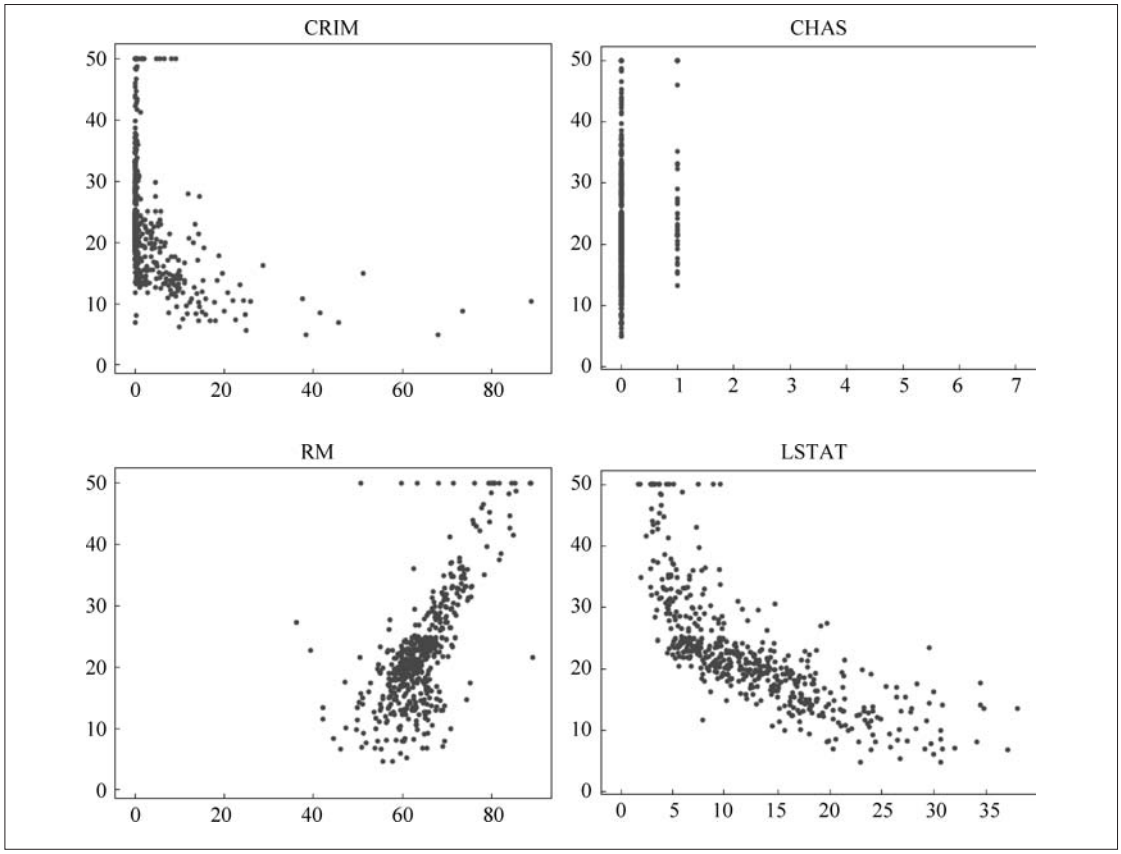
3. 异常数据处理

采用散点图来展示并分析数据。*X* 轴的值为每一个特征值，*Y* 轴是房价。

程序 3.7　波士顿数据集的散点图

```
1:   import pandas as pd
2:   import numpy as np
3:   import matplotlib.pyplot as plt
4:   from sklearn import datasets
5:   from sklearn.linear_model import LinearRegression
6:
7:   dataset = datasets.load_boston()
8:   x = dataset.data
9:   y = dataset.target
10:  names = dataset.feature_names
11:  fori in range(13):
12:      plt.plot(7,2,i + 1)
13:      plt.scatter(x[:,i],y,s = 10)
14:      plt.title(names[i])
15:      plt.show())
```

输出：有 13 个散点图，这里只展示其中的 4 个。



分析：观察 RM、LSTAT、PTRATIO 这三个散点图，Y 值为 50 点对应的 X 轴的值不同，并且很分散，可以判定为异常数据，考虑删除，其他值都较为正常。

经过以上两步，得到处理后的数据集。接下来通过使用这个数据集，并利用线性回归算法进行学习。

4. 线性回归分析

调用 sklearn.linear_model.LinearRegression()可实现线性回归分析，所需参数如下。

(1) fit_intercept：布尔型参数，表示是否计算该模型截距。可选参数。默认值为 True。

(2) normalize：布尔型参数，若为 True，则 X 在回归前进行归一化。可选参数。默认值为 False。

(3) copy_X：布尔型参数，若为 True，则 X 将被复制；否则将被覆盖。可选参数。默认值为 True。

(4) n_jobs：整型参数，表示用于计算的作业数量。若为－1，则用所有的 CPU。可选参数。默认值为 1。

程序 3.8　波士顿数据集线性回归分析

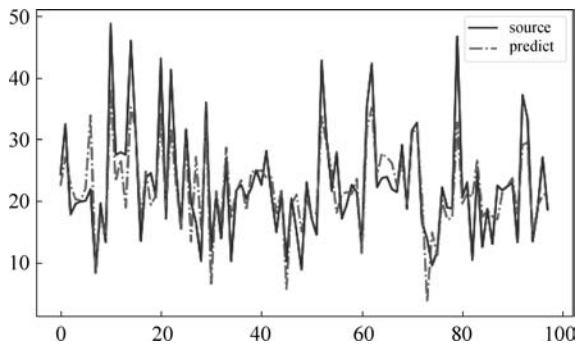
```
1:    import matplotlib.pyplot as plt
2:    from sklearn import datasets
3:    from sklearn.linear_model import LinearRegression
4:    import pandas as pd
```

```
from sklearn.model_selection import train_test_split
from pandas importDataFrame
from sklearn.metrics import r2_score

bos = datasets.load_boston()                    #获取数据
x = bos.data
y = bos.target
df = pd.DataFrame(x,columns = bos.feature_names)
features = ['CRIM', 'ZN', 'INDUS','CHAS','NOX','AGE','DIS','RAD','TAX','B']
tmp = df.drop(features,axis = 1)                #删除 features 存储的对应列
tmp_row = []                                    #存储删除的行号
fori in range(len(y)):
    if y[i] = = 50:
        tmp_row.append(i)                       #存储房价等于 50 的异常值下标
x = tmp.drop(tmp_row)
y = pd.DataFrame(y).drop(tmp_row)
#分割数据集
X_train,X_test,y_train,y_test = train_test_split(x,y,random_state = 0,test_size = 0.20)
print(len(X_train))
print(len(X_test))
lr = LinearRegression()
#使用训练数据进行参数估计
print(lr.intercept_)                            #截距
print(lr.coef_)                                 #线性模型的系数

lr.fit(X_train,y_train)
#回归预测
y_pred = lr.predict(X_test)
fig = plt.figure(figsize = (12, 6))
plt.plot(range(y_test.shape[0]), y_test, color = 'blue', linewidth = 1.5, linestyle = '-')
plt.plot(range(y_test.shape[0]), y_pred, color = 'red', linewidth = 1.5, linestyle = '-.')
plt.legend(["source", "predict "])
plt.show()
score = r2_score(y_test,y_pred)
print(score)
```

输出的曲线图形和数值：



```
392
98
[19.81059047]
[[ 3.88235108 -0.85618638 -0.51535387]]
  0.7062014880668344
```

分析：首先对数据集进行了处理，只保留 3 列数据，并删除了异常数据。将数据集分为训练集和测试集，数据集的 80%作为训练集，20%作为测试集。训练集有 392 条数据，测试集有 98 条数据。lr＝LinearRegression()获取线性回归函数，lr. fit(X_train，y_train)进行训练学习。[19.81059047]为线性模型的截距，[[3.88235108　－0.85618638　－0.51535387]]为线性模型的系数。然后以图形方式输出预测和实际数据对比图。从输出的曲线中可以清晰地看出，source 样式是原来的数据曲线，predict 为预测后的数据曲线。使用 R2_score 对模型评估，r2_score()函数计算 R 的平方，即确定系数，可以表示特征模型对特征样本预测的好坏，它的输出数值为 0.7062014880668344。

3.7　聚类

聚类是根据相似性原则，将具有较高相似度的数据对象划分至同一类簇，将具有较高相异度的数据对象划分至不同类簇。聚类与分类最大的区别在于，聚类过程为无监督过程，即待处理数据对象没有任何先验知识，而分类过程为有监督过程，即存在有先验知识的训练数据集。

聚类的目标是识别数据点的内在属性，使它们属于相同的子组。没有一种通用的相似性度量方法适用于所有情况。这取决于当前的问题。例如，可能对查找每个子组的代表性数据点感兴趣，或者对查找数据中的异常值感兴趣。根据情况，最终会选择合适的度量方法。

K-Means 算法是一种著名的数据聚类算法。该算法中的 K 代表类簇个数，K-Means 代表类簇内数据对象的均值(这种均值是一种对类簇中心的描述)，因此，K-Means 算法又称为 K-均值算法。K-Means 算法是一种基于划分的聚类算法，以距离作为数据对象间相似性度量的标准，即数据对象间的距离越小，则它们的相似性越高，它们越有可能在同一个类簇。数据对象间距离的计算有很多种，K-Means 算法通常采用欧氏距离来计算数据对象间的距离。

为了使用这个算法，需要假设集群的数量是预先知道的。然后使用不同的数据属性将数据分割成 K 个子组。首先确定集群的数量，并基于此对数据进行分类。这里的核心思想是，需要在每次迭代中更新这些 K 个质心的位置。继续迭代，直到将质心放置在它们的最佳位置。可见，质心的初始位置在算法中起着重要的作用。这些质心应该以一种巧妙的方式放置，因为这直接影响结果。一个好的策略是把它们尽可能地放在远离彼此的地方。

基本的 K-Means 算法将这些质心随机放置，接着从数据点的输入列表中根据算法来选择这些点。它试图把最初的质心彼此放置得很远，这样它就能很快地收敛。然后，遍历训练数据集，并将每个数据点都分配到离它最近的质心中去。一旦遍历完整个数据集，第一次迭代就结束了。算法已经根据初始化的质心对这些点进行了分组。现在，需要根据在第一次迭代结束时获得的新集群重新计算质心的位置。获得新的 K 个质心，需要再次重复上述过程，遍历数据集并将每个点都分配给最近的质心。

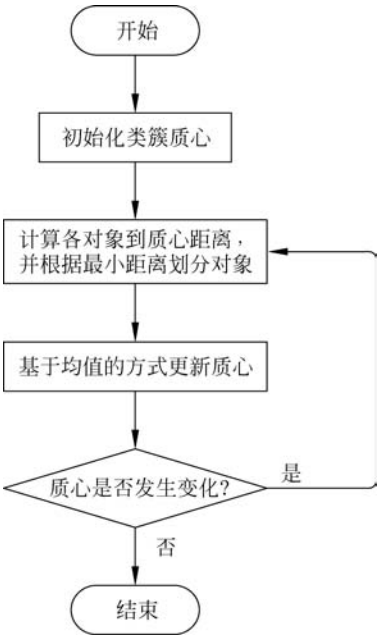


图 3-11　K-Means 聚类算法流程

当不断重复这些步骤时，质心会不断移动到它们的平衡位置。经过一定次数的迭代，质心不再改变它们的位置。这意味着质心已经到达了它的最终位置。最终生成的 K 个质心用于推断。

K-Means 聚类算法的具体步骤如下：

(1) 初始化质心。K-Means 算法需要事先确定类簇分支数，并初始化各类簇的质心。

(2) 聚类对象。K-Means 算法按照对象与质心间的距离划分类簇，其中，距离可以是欧式距离 $d_{\text{Euclidean}}$：

$$d_{\text{Euclidean}}=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$$

或是余弦距离 d_{cosine}：

$$d_{\text{cosine}}=\frac{x_1x_2+y_1y_2}{\sqrt{x_1^2+y_1^2}\sqrt{x_2^2+y_2^2}}$$

(3) 更新质心。K-Means 完成对象聚类后，计算各类簇中对象的平均值，并以此作为新的质心。

梳理算法的脉络，可构建出一个完整的 K-Means 聚类算法流程，如图 3-11 所示。

11 K-Means 算法演示

下面实现 K-Means 算法。

程序 3.9　K-Means 算法

```
1:  import numpy as np
2:  import matplotlib.pyplot as plt
3:  from sklearn.cluster import KMeans
4:
5:  X = np.loadtxt('data_clustering.txt', delimiter = ',')
6:  num_clusters = 5
7:
8:  plt.figure()
9:  plt.scatter(X[:,0], X[:,1], marker = 'o', facecolors = 'none',
10:           edgecolors = 'black', s = 80)
11: x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
12: y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
13: plt.title('Input data')
14: plt.xlim(x_min, x_max)
15: plt.ylim(y_min, y_max)
16: plt.xticks(())
17: plt.yticks(())
18:
19: kmeans = KMeans(init = 'k - means + + ', n_clusters = num_clusters, n_init = 10)
20: kmeans.fit(X)
21:
22: step_size = 0.01
23: x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
```

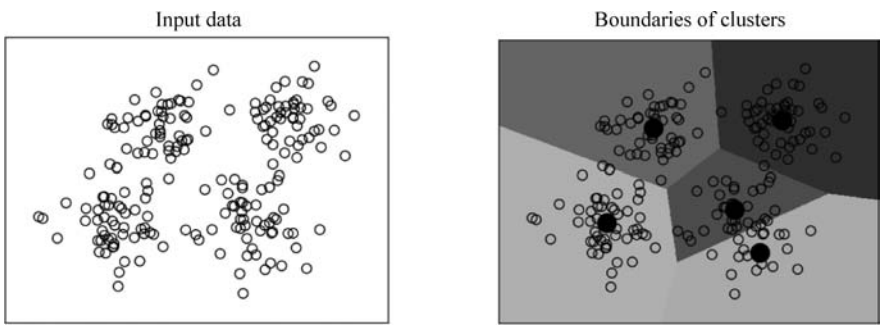
```
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
x_vals, y_vals = np.meshgrid(np.arange(x_min, x_max, step_size),
                             np.arange(y_min, y_max, step_size))

output = kmeans.predict(np.c_[x_vals.ravel(), y_vals.ravel()])
output = output.reshape(x_vals.shape)

plt.figure()
plt.clf()
plt.imshow(output, interpolation='nearest', extent=(x_vals.min(),
       x_vals.max(), y_vals.min(), y_vals.max()),
       cmap=plt.cm.Paired, aspect='auto', origin='lower')
plt.scatter(X[:,0], X[:,1], marker='o', facecolors='none',
       edgecolors='black', s=80)

cluster_centers = kmeans.cluster_centers_
plt.scatter(cluster_centers[:,0], cluster_centers[:,1],
       marker='o', s=210, linewidths=4, color='black',
       zorder=12, facecolors='black')
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
plt.title('Boundaries of clusters')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```

输出：



分析：首先从 sklearn 库中导入聚类模块 KMeans，从 data_clustering.txt 文件中加载源数据，并定义好集群的数量，这里集群数量定义为 5。接着对输入数据进行可视化，第一幅图展示的是输入数据。可以直观地看到在这个数据中有五个分组。使用初始化参数创建 K-Means 对象。init 参数表示选择集群初始中心的初始化方法。使用 K-Means++以更智能的方式选择这些中心，而不是随机选择它们。这保证了算法的快速收敛。n_clusters 参数表示集群的数量。n_init 参数是指算法在确定最佳结果之前应该运行的次数。接着用输入数据对 K-Means 模型进行训练。最后绘图将训练结果进行可视化。第二幅图展示了训练后的结

果,它成功地将输入数据分为了五个区域,并为每个簇的中心用黑点标了出来。

3.8 小结

在本章中,首先了解了什么是机器学习,然后对机器学习做了分类,学习了监督学习、半监督学习和非监督学习的区别。接着学习了逻辑回归的概念,并用它们构建了分类器。然后学习了线性回归,并解决了房价预测问题。最后学习了聚类算法并用程序实现了 K-Means 算法。机器学习算法还有很多,本章只是简单地介绍几种算法及其应用,有了这些基础,机器学习的基本流程就清楚了,可以进一步深入学习。

习题

1. 谈谈你对机器学习的理解,包括回归和分类的相同点和不同点。
2. 简述机器学习的流程。
3. 简述监督学习与无监督学习之间的区别。
4. 数据预处理过程中,对于异常数据处理的方法有哪些?
5. 实现本章逻辑回归分类、线性回归和聚类的实例。
6. 软聚类通过约束来放宽聚类的边界,从而解决重叠聚类、离群点和不确定对象的问题(也就是说一个对象可以属于多个聚类)。作为一种典型软聚类方法,三支聚类获得了众多的关注。请尝试实现三支 K-Means 算法。