第 1 章 机器学习概述

CHAPTER 1

机器学习是计算机科学与统计学结合的产物，主要研究如何选择统计学习模型，从大量已有数据中学习特定经验。机器学习中的经验称为模型，机器学习的过程即根据一定的性能度量准则对模型参数进行近似求解，以使得模型在面对新数据时能够给出相应的经验指导。对于机器学习的准确定义，目前学术界尚未有统一的描述，比较常见的是 Mitchell 教授[1]于 1997 年对机器学习的定义："对于某类任务 T 和性能度量 P，一个计算机程序被认为可以从经验 E 中学习是指：通过经验 E 改进后，它在任务 T 上的性能度量 P 有所提升。"[2]

1.1 机器学习的组成

对于一个给定数据集，使用机器学习算法对其进行建模，以学习其中的经验。构建一个完整的机器学习算法需要三个方面的要素，分别是数据、模型、性能度量准则。

首先是数据方面。数据是机器学习算法的原材料，其中蕴含了数据的分布规律。生产实践中直接得到的一线数据往往是"脏数据"，可能包含大量缺失值、冗余值，而且不同维度获得数据的量纲往往也不尽相同。对于这样的"脏数据"，通常需要先期的特征工程进行预处理。

其次是模型方面。如何从众多机器学习模型中选择一个来对数据建模，是一个依赖于数据特点和研究人员经验的问题。常见的机器学习算法主要有：逻辑回归、最大熵模型、k-近邻模型、决策树、朴素贝叶斯分类器、支持向量机、高斯混合模型、隐马尔可夫模型、降维、聚类、深度学习等。特别是近些年来深度学习领域的发展，给产业界带来了一场智能化革命，各行各业纷纷使用深度学习进行行业赋能。

最后是性能度量准则。性能度量准则用于指导机器学习模型进行模型参数求解。这一参数求解过程称为训练，训练的目的是使性能度量准则在给定数据集上达到最优。训练一个机器学习模型往往需要对大量的参数进行反复调整或者搜索，这一过程称为调参。其中，在训练之前调整设置的参数，称为超参数。

按照不同的划分准则，机器学习算法可以分为不同的类型。下面介绍几种常见的机器学习算法划分方法。

1.2 分类问题及回归问题

根据模型预测输出的连续性，可以将机器学习算法适配的问题划分为分类问题和回归问题。分类问题以离散随机变量或者离散随机变量的概率分布作为输出，回归问题以连续变量作为预测输出。分类模型的典型应用有：图像分类、视频分类、文本分类、机器翻译、语音识别等。回归模型的典型应用有：银行信贷评分、人脸/人体关键点估计、年龄估计、股市预测等。

在某些情况下，回归问题与分类问题之间可以相互转化。例如对于估计人的年龄问题，假设绝大多数人的年龄介于 0 到 100 岁，那么可以将年龄估计问题看作是一个 0～100 之间实数的回归问题，也可以将其量化为一个 101 个年龄类别的分类问题。

1.3 监督学习、半监督学习和无监督学习

根据样本集合中是否包含标签以及包含标签的多少，可以将机器学习分为监督学习、半监督学习和无监督学习。

监督学习是指样本集合中包含标签的机器学习。给定有标注的数据集 $D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_m,y_m)\}$，以$\{y_1,y_2,\cdots,y_m\}$作为监督信息，来最小化损失函数 J，通过梯度下降、拟牛顿法等算法来对模型的参数进行更新。其中，损失函数 J 用于描述模型的预测值与真实值之间的差异度，差异度越小，模型对数据拟合效果越好。

然而获得有标注的样本集合往往需要耗费大量的人力、财力。有时我们也希望能够从无标注数据中发掘出新的信息，比如电商平台根据用户的特征对用户进行归类，以实现商品的精准推荐，这时就需要用到无监督学习。降维、聚类是最典型的无监督学习算法。

半监督学习介于监督学习和无监督学习之间。在某些情况下，我们仅能够获得部分样本的标签。半监督学习就是同时从有标签数据及无标签数据中进行经验学习的机器学习。

1.4 生成模型及判别模型

根据机器学习模型是否可用于生成新数据，可以将机器学习模型分为生成模型和判别模型。生成模型是指通过机器学习算法，从训练集中学习到输入和输出的联合概率分布 $P(X,Y)$。对于新给定的样本，计算 X 与不同标记之间的联合分布概率，选择其中最大的概率对应的标签作为预测输出。典型的生成模型有：朴素贝叶斯分类器、高斯混合模型、隐马尔可夫模型、生成对抗网络等。而判别模型计算的是一个条件概率分布 $P(X,Y)$，即后验概率分布。典型的判别模型有逻辑回归、决策树、支持向量机、神经网络、k-近邻算法。由于生成模型学习的是样本输入与标签的联合概率分布，所以我们可以从生成模型的联合概率分布中进行采样，从而生成新的数据，而判别模型只是一个条件概率分布模型，只能对输入进行判定。

1.5　模型评估

1.5.1　训练误差及泛化误差

对于给定的一批数据，要求使用机器学习对其进行建模。通常首先将数据划分为训练集、验证集和测试集三个部分。训练集用于对模型的参数进行训练；验证集用于对训练的模型进行验证挑选、辅助调参；测试集用于测试训练完成的模型的泛化能力。在训练集上，训练过程中使用训练误差来衡量模型对训练数据的拟合能力，而在测试集上则使用泛化误差来测试模型的泛化能力。在模型得到充分训练的条件下，训练误差与泛化误差之间的差异越小，说明模型的泛化性能越好，得到一个泛化性能好的模型是机器学习的目的。训练误差和测试误差往往选择的是同一性能度量函数，只是作用的数据集不同。

1.5.2　过拟合及欠拟合

当训练损失较大时，说明模型不能对数据进行很好的拟合，称这种情况为欠拟合。当训练误差小且明显低于泛化误差时，称这种情况为过拟合，此时模型的泛化能力往往较弱。如图 1-1 所示，图中的样本点围绕曲线 $y=x^2$ 随机采样而得，当使用二次多项式对样本点进行拟合时可以得到曲线 $y=1.01x^2+0.0903x-3.75$，尽管几乎所有的样本点都不在该曲线上，但该方程与 $y=x^2$ 整体上重合，因此拟合效果较好；当采用一次多项式进行拟合时可以得到曲线 $y=9.211x-15.91$，此时在样本集合上多项式都不能对数据进行很好的拟合，模型对数据欠拟合；而使用五次多项式对样本点进行拟合时，得到的多项式为 $y=-0.00343x^5+0.0185x^4+0.590x^3-4.86x^2+15.0x-9.58$，曲线几乎通过了每个样本点，但是当 $x>10$ 时，则会发生明显的预测错误（泛化能力弱），模型对数据过拟合。

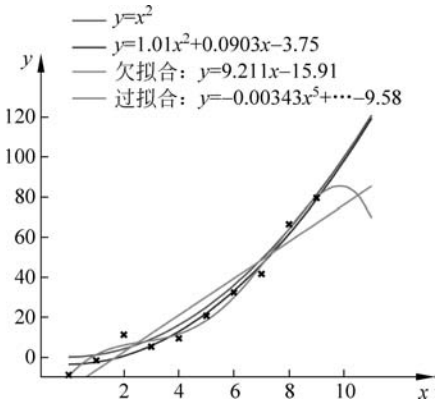

图 1-1　过拟合及欠拟合（见彩插）

对于欠拟合的情况，通常是由于模型本身不能对训练集进行拟合或者训练迭代次数太少。在图 1-1 中，线性模型不能近似拟合二次函数。解决欠拟合的主要方法是对模型进行改进、设计新的模型重新训练、增加训练过程的迭代次数等。对于过拟合的情况，往往是由于数据量太少或者模型太复杂导致，可以通过增加训练数据量、对模型进行裁剪、正则化等方式来缓解。

1.6　正则化

正则化（Normalization）是一种抑制模型复杂度的常用方法。正则化用模型参数 $\boldsymbol{\omega}$ 的 p 范数进行表示为

$$\| \boldsymbol{\omega} \|_p = \left(\sum_{i=1}^{p} | \omega_i |^p \right)^{\frac{1}{p}} \tag{1-1}$$

常用的正则化方式为 $p=1$ 或 $p=2$ 的情形，分别称为 L1 正则化和 L2 正则化。正则化项一般作为损失函数的一部分被加入到原来的基于数据损失函数当中。基于数据的损失函数又称为经验损失，正则化项又称为结构损失。若将原本基于数据的损失函数记为 J，带有正则化项的损失函数记为 J_N，则最终的损失函数可记为

$$J_N = J + \lambda \| \boldsymbol{\omega} \|_p \tag{1-2}$$

其中，λ 是用于在模型的经验损失和结构损失之间进行平衡的超参数。

L1 正则化是模型参数的 1 范数。以图 1-2 为例，假设某个模型参数只有(ω_1, ω_2)，P 点为其训练集上的全局最优解。在没有引入正则项时，模型很可能收敛到 P 点，从而引发严重的过拟合。正则项的引入会迫使参数的取值向原点方向移动，从而减轻了模型过拟合的程度。对于图 1-2，当$|\omega_1|+|\omega_2|$固定时，损失函数在$\boldsymbol{\omega}^*$处取得最小值。此时 $\omega_1=0$，因此与 ω_1 对应的“特征分量”在决策中将不起作用，这时称模型获得了“稀疏”解。对于图 1-2，模型的损失函数等值线为圆形的特殊情况，模型能够取得“稀疏解”的条件是其全局最优解落在图中的阴影区域。更一般的 L1 正则化能够以较大的概率获得稀疏解，起到特征选择的作用。需要注意的是，L1 正则化可能得到不止一个最优解。

L2 正则化是模型参数的 2 范数。从图 1-3 中可以看到，对于模型的损失函数的等值线是圆的特殊情况，仅当等值线与正则化损失的等值线相切时，模型才能获得“稀疏”解。与 L1 正则化相比，获得“稀疏”解的概率要小得多，故 L2 正则化得到的解更加平滑。

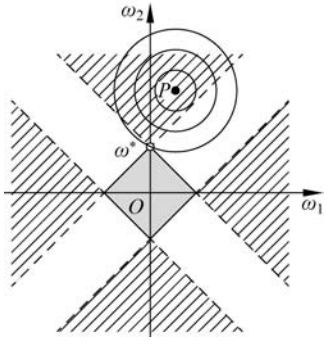

图 1-2　L1 正则化

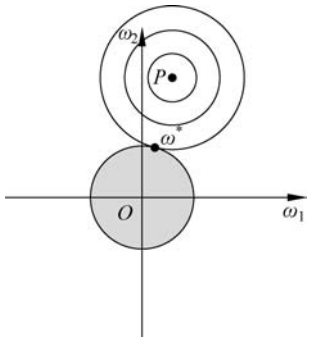

图 1-3　L2 正则化

可以看到，存在多个解可选时，L1 和 L2 正则化都能使参数尽可能地靠近零，这样得到的模型会更加简单。实际应用当中，由于 L2 正则化有着良好的数学性质，在计算上更加方便，所以人们往往选择 L2 正则化来防止过拟合。

1.7　Scikit-learn 模块

Scikit-learn 简称 sklearn，是 Python 中常用的机器学习模块。sklearn 封装了许多机器学习方法，例如数据预处理、交叉验证等。除模型部分外，本节对一些常用 API 进行简要介绍，以便读者理解后文中的实例。模型部分的 API 会在相关章节进行介绍。

1.7.1 数据集

sklearn.datasets 中收录了一些标准数据集，例如鸢尾花数据集、葡萄酒数据集等。这些数据集通过一系列 load 函数加载，例如 sklearn.datasets.load_iris 函数可以加载鸢尾花数据集。load 函数的返回值是一个 sklearn.utils.Bunch 类型的变量，其中最重要的成员是 data 和 target，分别表示数据集的特征和标签。代码清单 1-1 展示了加载鸢尾花数据集的方法，其他数据集的加载方式与之类似，请读者自行尝试。

代码清单 1-1 加载数据集

```
from sklearn.datasets import load_iris
iris = load_iris()
x = iris.data
y = iris.target
```

鸢尾花数据集(Iris Data Set)是统计学和机器学习中常被当作示例使用的一个经典的数据集。该数据集共 150 个样本，分为 Setosa、Versicolour 和 Virginica 共 3 个类别。每个样本用四个维度的属性进行描述：分别是用厘米(cm)表示的花萼长度(Sepal Length)、花萼宽度(Sepal Width)、花瓣长度(Petal Length)和花瓣宽度(Petal Width)。

葡萄酒数据集(Wine Data Set)包含 178 条记录，来自 3 种不同起源地。数据集的 13 个属性是葡萄酒的 13 种化学成分，包括 Alcohol、Malic acid、Ash、Alcalinity of ash、Magnesium、Total phenols、Flavanoids、Nonflavanoid phenols、Proanthocyanins、Color intensity、Hue、OD280/OD315 of diluted wines、Proline。

波士顿房价数据集(Boston Data Set)从 1978 年开始统计，共包含 506 条数据。样本标签为平均房价，13 个特征包括城镇人均犯罪率(CRIM)、房间数(RM)等。由于样本标签为连续变量，所以波士顿房价数据集可以用于回归模型。图 1-4 绘制了各个特征与标签之间的关系。可以发现，除了 CHAS 和 RAD 特征外，其他特征均与结果呈现出较高的相关性。

乳腺癌数据集(Breast Cancer Data Set)一共包含 569 条数据，其中有 357 例乳腺癌数据以及 212 例非乳腺癌数据。数据集中包含 30 个特征，这里不一一罗列。有兴趣的读者可以使用代码清单 1-2 查询详细信息。

代码清单 1-2 乳腺癌数据集详细信息

```
from sklearn.datasets import load_breast_cancer
bc = load_breast_cancer()
print(bc.DESCR)
```

1.7.2 模型选择

sklearn.model_selection 中提供了有关模型选择的一系列工具，包括验证集划分、交叉验证等。验证集与训练集的划分是所有项目都需要使用的，因此本节主要介绍这一 API。其他功能会在使用时加以讲解。

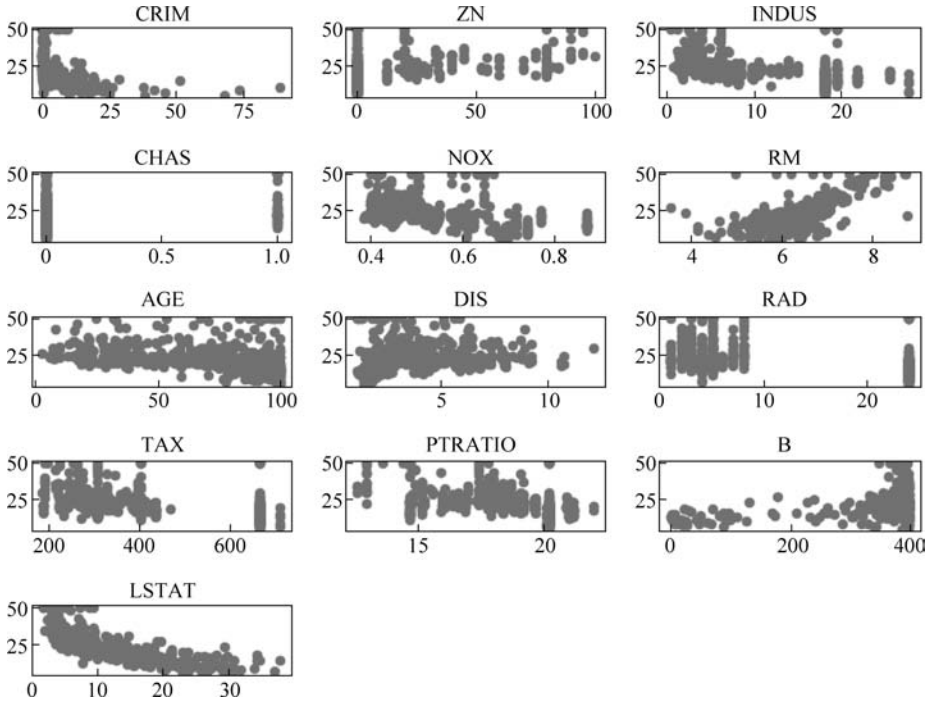

图 1-4　波士顿房价数据集各特征与标签之间的关系(见彩插)

验证集的划分主要通过 train_test_split(* arrays， ** options)函数实现。参数 arrays 包含待划分的数据,其中每个元素都是长度相同的列表。验证集划分的目标是将这些列表划分为两段,一段作为训练集,一段作为验证集。关键字参数包括 test_size、shuffle 等。其中 test_size 规定了测试集占完整数据集的比例,默认取 0.25。shuffle 选项决定数据集是否被打乱,默认值为 True。代码清单 1-3 展示了 train_test_split 函数的常见使用方法。

代码清单 1-3　train_test_split 函数的常见用法

```
x_train, x_test, y_train, y_test = train_test_split(data, target)
```

第 2 章 逻辑回归及最大熵模型

CHAPTER 2

逻辑回归模型是一种常用的回归或分类模型，可以视为广义线性模型的特例。本节首先给出线性回归模型和广义线性模型的概念，然后介绍逻辑回归和多分类逻辑回归。随后，本节还将介绍如何通过最大熵模型解释逻辑回归。

2.1 线性回归

线性回归是最基本的回归分析方法，有着广泛的应用。线性回归研究的是自变量与因变量之间的线性关系。对于特征 $\boldsymbol{x}=(x^1,x^2,\cdots,x^n)$ 及其对应的标签 y，线性回归假设二者之间存在线性映射

$$y \approx f(x)=\omega_1 x^1+\omega_2 x^2+\cdots+\omega_n x^n+b=\sum_{i=1}^{n}\omega_i x^i+b=\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x}+b \tag{2-1}$$

其中，$\boldsymbol{\omega}=(\omega_1,\omega_2,\cdots,\omega_n)$ 和 b 分别表示待学习的权重及偏置。直观上，权重 $\boldsymbol{\omega}$ 的各个分量反映了每个特征变量的重要程度。权重越大，对应的随机变量的重要程度越大，反之则越小。

线性回归的目标是求解 $\boldsymbol{\omega}$ 和 b，使得 $f(\boldsymbol{x})$ 与 y 尽可能接近。求解线性回归模型的基本方法是最小二乘法。最小二乘法是一个不带条件的最优化问题，优化目标是让整个样本集合上的预测值与真实值之间的欧氏距离之和最小。

2.1.1 一元线性回归

式(2-1)描述的是多元线性回归。为简化讨论，首先以一元线性回归为例进行说明

$$y \approx f(x)=\omega x+b \tag{2-2}$$

给定空间中的一组样本点 $D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)\}$，目标函数为

$$J(\omega,b)=\sum_{i=1}^{m}(y_i-f(x_i))^2=\sum_{i=1}^{m}(y_i-\omega x_i-b)^2 \tag{2-3}$$

令目标函数对 ω 和 b 的偏导数为 0：

$$\begin{cases}\dfrac{\partial J(\omega,b)}{\partial \omega}=\displaystyle\sum_{i=1}^{m}2\omega x_i^2+\sum_{i=1}^{m}2(b-y_i)x_i=0\\[2ex]\dfrac{\partial J(\omega,b)}{\partial b}=\displaystyle\sum_{i=1}^{m}2(\omega x_i-y_i)+2mb\end{cases} \tag{2-4}$$

则可得到 ω 和 b 的估计值为

$$\omega=\frac{m\sum_{i=1}^{m}x_iy_i-\sum_{i=1}^{m}x_i\sum_{i=1}^{m}y_i}{m\sum_{i=1}^{m}x_i^2-(\sum_{i=1}^{m}x_i)^2}=\frac{\overline{xy}-\bar{x}\cdot\bar{y}}{\bar{x}^2-\bar{x}^2}$$

$$b=\frac{1}{m}(\sum_{i=1}^{m}y_i-\omega\sum_{i=1}^{m}x_i)=\bar{y}-\omega\bar{x} \tag{2-5}$$

其中，短横线“ $^-$ ”表示求均值运算。

2.1.2 多元线性回归

对于多元线性回归，本书仅做简单介绍。为了简化说明，可以将 b 同样看作权重，即令

$$\begin{cases}\boldsymbol{\omega}=(\omega_1,\omega_2,\cdots,\omega_n,b)\\ \boldsymbol{x}=(x^1,x^2,\cdots,x^n,1)\end{cases} \tag{2-6}$$

此时式(2-1)可表示为

$$y\approx f(x)=\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x} \tag{2-7}$$

给定空间中的一组样本点 $D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_m,y_m)\}$，优化目标为

$$\min J(\boldsymbol{\omega})=\min(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\omega})^{\mathrm{T}}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\omega}) \tag{2-8}$$

其中，$\boldsymbol{X}$ 为样本矩阵的增广矩阵

$$\boldsymbol{X}=\begin{pmatrix}x_1^1 & x_1^2 & \cdots & x_1^n & 1\\ x_2^1 & x_2^2 & \cdots & x_2^n & 1\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ x_m^1 & x_m^2 & \cdots & x_m^n & 1\end{pmatrix} \tag{2-9}$$

$\boldsymbol{Y}$ 为对应的标签向量

$$\boldsymbol{Y}=(y_1,y_2,\cdots,y_n)^{\mathrm{T}} \tag{2-10}$$

求解式(2-8)可得

$$\boldsymbol{\omega}=(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{Y} \tag{2-11}$$

当 $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ 可逆时，线性回归模型存在唯一解。当样本集合中的样本太少或者存在大量线性相关的维度，则可能会出现多个解的情况。奥卡姆剃刀原则指出，当模型存在多个解时，选择最简单的那个。因此可以在原始线性回归模型的基础上增加正则化项目以降低模型的复杂度，使得模型变得简单。若加入 L2 正则化，则优化目标可写作

$$\min J(\boldsymbol{\omega})=\min(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\omega})^{\mathrm{T}}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\omega})+\lambda\|\boldsymbol{\omega}\|_2 \tag{2-12}$$

此时，线性回归又称为岭(Ridge)回归。求解式(2-12)有

$$\boldsymbol{\omega}=(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}+\lambda\boldsymbol{I})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{Y} \tag{2-13}$$

$\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}+\lambda\boldsymbol{I}$ 在 $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ 的基础上增加了一个扰动项 $\lambda\boldsymbol{I}$。此时不仅能够降低模型的复杂度、防止过拟合，而且能够使 $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}+\lambda\boldsymbol{I}$ 可逆，$\boldsymbol{\omega}$ 有唯一解。

当正则化项为 L1 正则化时，线性回归模型又称为 Lasso(Least Absolute Shrinkage and Selection Operator)回归，此时优化目标可写作

$$\min J(\boldsymbol{\omega}) = \min(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\omega})^{\mathrm{T}}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\omega}) + \lambda \|\boldsymbol{\omega}\|_1 \tag{2-14}$$

L1 正则化能够得到比 L2 正则化更为稀疏的解。如 1.6 节，稀疏是指$\boldsymbol{\omega} = (\omega_1, \omega_2, \cdots, \omega_n)$中会存在多个值为 0 的元素，从而起到特征选择的作用。由于 L1 范数使用绝对值表示，所以目标函数 $J(\boldsymbol{\omega})$不是连续可导，此时不能再使用最小二乘法进行求解，可使用近端梯度下降进行求解(PGD)，本书略。

线性模型通常是其他模型的基本组成单元。堆叠若干个线性模型，同时引入非线性化激活函数，就可以实现对任意数据的建模。例如，神经网络中的一个神经元就是由线性模型加激活函数组合而成。

2.2　广义线性回归

上面描述的都是狭义线性回归，其基本假设是 y 与 x 直接呈线性关系。如果 y 与 x 不是线性关系，那么使用线性回归模型进行拟合后会得到较大的误差。为了解决这个问题，可以寻找这样一个函数 $g(y)$，使得 $g(y)$与 x 之间是线性关系。举例来说，假设 x 是一个标量，y 与 x 的实际关系是 $y = \omega x^3$。令

$$g(y) = y^{1/3} = \omega' x \tag{2-15}$$

其中，$\omega' = \omega^{1/3}$ 是要估计的未知参数。那么 $g(y)$与 x 呈线性关系，此时可以使用线性回归对 ω'进行参数估计，从而间接得到 ω。这样的回归称为广义线性回归。实际场景中，g 的选择是最关键的一步，一般较为困难。

2.2.1　逻辑回归

逻辑回归是一种广义线性回归，通过回归对数几率(Logits)的方式将线性回归应用于分类任务。对于一个二分类问题，令 $Y \in \{0,1\}$表示样本 x 对应的类别变量。设 x 属于类别 1 的概率为 $P(Y=1|x) = p$，则自然有 $P(Y=0|x) = 1-p$。比值 $\dfrac{p}{1-p}$ 称为几率(Odds)，几率的对数即为对数几率(Logits)

$$\ln \frac{p}{1-p} \tag{2-16}$$

逻辑回归通过回归式(2-16)来间接得到 p 的值，即

$$\ln \frac{p}{1-p} = \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x} + b \tag{2-17}$$

解得

$$p = \frac{1}{1 + \mathrm{e}^{-(\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x} + b)}} \tag{2-18}$$

为方便描述，令

$$\begin{cases} \boldsymbol{\omega} = (\omega_1, \omega_2, \cdots, \omega_n, b)^{\mathrm{T}} \\ \boldsymbol{x} = (x^1, x^2, \cdots, x^n, 1)^{\mathrm{T}} \end{cases} \tag{2-19}$$

则有

$$p=\frac{1}{1+\mathrm{e}^{-\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x}}} \tag{2-20}$$

由于样本集合给定的样本属于类别1的概率非0即1，所以式(2-20)无法用最小二乘法求解。此时可以考虑使用极大似然估计进行求解。

给定样本集合 $D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_m,y_m)\}$，似然函数为

$$L(\boldsymbol{\omega})=\prod_{i=1}^{m}p^{y_i}(1-p)^{(1-y_i)} \tag{2-21}$$

对数似然函数为

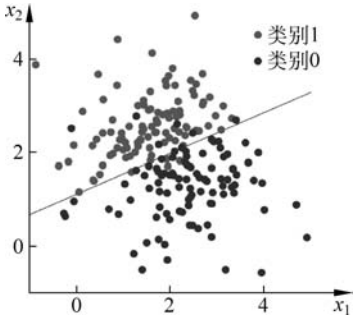

图 2-1　逻辑回归示例(见彩插)

$$\begin{aligned}l(\boldsymbol{\omega})&=\sum_{i=1}^{m}(y_i\ln p+(1-y_i)\ln(1-p))\\&=\sum_{i=1}^{m}(y_i\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x}_i-\ln(1+\mathrm{e}^{\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x}_i}))\end{aligned} \tag{2-22}$$

之后可用经典的启发式最优化算法梯度下降法(见11.4节)求解式(2-22)。

图2-1是二维空间中使用逻辑回归进行二分类的示例。图中样本存在一定的噪声(正类中混合有部分负类样本、负类中混合有部分正类样本)。可以看到逻辑回归能够抵御一定的噪声干扰。

2.2.2　多分类逻辑回归

二分类逻辑回归也可扩展到多分类逻辑回归。

将 $\boldsymbol{\omega}=\boldsymbol{\omega}_1-\boldsymbol{\omega}_0$ 带入式(2-20)有

$$\begin{cases}P(Y=1\mid\boldsymbol{x})=p=\dfrac{1}{1+\mathrm{e}^{-\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x}}}\\\qquad\qquad=\dfrac{\mathrm{e}^{\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x}}}{1+\mathrm{e}^{\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x}}}=\dfrac{\mathrm{e}^{\boldsymbol{\omega}_1^{\mathrm{T}}\boldsymbol{x}}}{\mathrm{e}^{\boldsymbol{\omega}_0^{\mathrm{T}}\boldsymbol{x}}+\mathrm{e}^{\boldsymbol{\omega}_1^{\mathrm{T}}\boldsymbol{x}}}\\P(Y=0\mid\boldsymbol{x})=1-p=\dfrac{\mathrm{e}^{\boldsymbol{\omega}_0^{\mathrm{T}}\boldsymbol{x}}}{\mathrm{e}^{\boldsymbol{\omega}_0^{\mathrm{T}}\boldsymbol{x}}+\mathrm{e}^{\boldsymbol{\omega}_1^{\mathrm{T}}\boldsymbol{x}}}\end{cases} \tag{2-23}$$

通过归纳可将逻辑回归推广到任意多分类问题中。当类别数目为 K 时(假设类别编号为 $0,1,\cdots,K-1$)有

$$P(Y=i\mid\boldsymbol{x})=\frac{\exp(\boldsymbol{\omega}_i^{\mathrm{T}}\boldsymbol{x})}{\sum_{k=0}^{K-1}\exp(\boldsymbol{\omega}_k^{\mathrm{T}}\boldsymbol{x})},\quad i=0,1,2,\cdots,K-1 \tag{2-24}$$

令式(2-24)的分子分母都除以 $\exp(\boldsymbol{\omega}_0\boldsymbol{x})$，则有

$$P(Y=0\mid\boldsymbol{x})=\frac{1}{1+\sum_{k=1}^{K-1}\mathrm{e}^{(\boldsymbol{\omega}_k^{\mathrm{T}}-\boldsymbol{\omega}_0^{\mathrm{T}})\boldsymbol{x}}}$$

$$P(Y=i \mid \boldsymbol{x})=\frac{\mathrm{e}^{(\boldsymbol{\omega}_i^{\mathrm{T}}-\boldsymbol{\omega}_0^{\mathrm{T}})\boldsymbol{x}}}{1+\sum_{k=1}^{K-1}\mathrm{e}^{(\boldsymbol{\omega}_k^{\mathrm{T}}-\boldsymbol{\omega}_0^{\mathrm{T}})\boldsymbol{x}}},\quad i=1,2,\cdots,K-1 \tag{2-25}$$

式(2-25)同样可以通过极大似然估计的方式转化成对数似然函数,然后通过梯度下降法求解。

2.2.3　交叉熵损失函数

交叉熵损失函数是神经网络中常用的一种损失函数。K 分类问题中,假设样本 $\boldsymbol{x}_i$ 属于每个类别的真实概率为 $\boldsymbol{p}_i=\{p_i^0,p_i^1,\cdots,p_i^{K-1}\}$,其中只有样本所属的类别的位置值为1,其余位置皆为0。假设分类模型的参数为$\boldsymbol{\omega}$,其预测的样本 $\boldsymbol{x}_i$ 属于每个类别的概率 $\boldsymbol{q}=\{q_i^0,q_i^1,\cdots,q_i^{K-1}\}$满足

$$\sum_{k=0}^{K-1}q_i^k=1 \tag{2-26}$$

则样本 $\boldsymbol{x}_i$ 的交叉熵损失定义为

$$J_i(\boldsymbol{\omega})=-\sum_{k=0}^{K-1}p_i^k\ln q_i^k \tag{2-27}$$

对所有样本有

$$J(\boldsymbol{\omega})=\sum_{i=1}^{m}J_i(\boldsymbol{\omega})=-\sum_{i=1}^{m}\sum_{k=0}^{K-1}p_i^k\ln q_i^k \tag{2-28}$$

当 $K=2$ 时,式(2-28)与式(2-22)形式相同。所以交叉熵损失函数与通过极大似然函数导出的对数似然函数类似,可以通过梯度下降法求解。

2.3　最大熵模型

信息论中,熵可以度量随机变量的不确定性。现实世界中,不加约束的事物都会朝着“熵增”的方向发展,也就是向不确定性增加的方向发展。可以证明,当随机变量呈均匀分布时,熵值最大。不仅在信息论中,在物理学、化学等领域中,熵都有着重要的应用。一个有序的系统有着较小的熵值,而一个无序系统的熵值则较大。

机器学习中,最大熵原理即假设:描述一个概率分布时,在满足所有约束条件的情况下,熵最大的模型是最好的。这样的假设符合“熵增”的客观规律,即在满足所有约束条件下,数据是随机分布的。以企业的管理条例为例,一般的管理条例规定了员工的办事准则,而对于管理条例中未规定的行为,在可供选择的选项中,员工们会有不同的选择。可以认为每个选项被选中的概率是相等的。实际情况也往往如此,这就是一个熵增的过程。

对于离散随机变量 x,假设其有 M 个取值。记 $p_i=P(x=i)$,则其熵定义为

$$H(P)=-\sum_{i=1}^{M}p_i\ln p_i \tag{2-29}$$

对于连续变量 x,假设其概率密度函数为 $f(x)$,则其熵定义为

$$H(f)=\int f(x)\ln f(x)\mathrm{d}x \tag{2-30}$$

2.3.1 最大熵模型的导出

给定一个大小为 m 的样本集合 $D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_m,y_m)\}$，假设输入变量为 $\boldsymbol{X}$，输出变量为 Y。以频率代替概率，可以估计出 $\boldsymbol{X}$ 的边缘分布及$(\boldsymbol{X},Y)$的联合分布为

$$\begin{cases}\tilde{p}(\boldsymbol{x},y)=\dfrac{N_{x,y}}{m}\\ \tilde{p}(\boldsymbol{x})=\dfrac{N_x}{m}\end{cases} \tag{2-31}$$

其中，$N_{x,y}$ 和 N_x 分别表示训练样本中$(\boldsymbol{X}=\boldsymbol{x},Y=y)$出现的频数和 $\boldsymbol{X}=\boldsymbol{x}$ 出现的频数。在样本量足够大的情况下，认为 $\tilde{p}(\boldsymbol{x})$反映真实的样本分布。基于此，最大熵模型使用条件熵进行建模，而非最大熵原理中一般意义上的熵。这样间接起到了缩小模型假设空间的作用。

$$H(p)=-\sum_{(\boldsymbol{x},y)\in D}\tilde{p}(\boldsymbol{x})P(y\mid\boldsymbol{x})\log P(y\mid\boldsymbol{x}) \tag{2-32}$$

根据定义，最大熵模型是在满足一定约束条件下熵最大的模型。最大熵模型的思路是：从样本集合使用特征函数 $f(\boldsymbol{x},y)$抽取特征，然后希望特征函数 $f(\boldsymbol{x},y)$关于经验联合分布 $\tilde{p}(\boldsymbol{x},y)$的期望，等于特征函数 $f(\boldsymbol{x},y)$关于模型 $p(y|\boldsymbol{x})$和经验边缘分布 $\tilde{p}(\boldsymbol{x})$的期望。

特征函数关于经验联合分布 $\tilde{p}(\boldsymbol{x},y)$的期望定义为

$$E_{\tilde{p}}(f)=\sum_{(\boldsymbol{x},y)\in D}\tilde{p}(\boldsymbol{x},y)f(\boldsymbol{x},y) \tag{2-33}$$

特征函数 $f(\boldsymbol{x},y)$关于模型 $p(y|\boldsymbol{x})$和经验边缘分布 $\tilde{p}(\boldsymbol{x})$的期望定义为

$$E_p(f)=\sum_{(\boldsymbol{x},y)\in D}\tilde{p}(\boldsymbol{x})p(y\mid\boldsymbol{x})f(\boldsymbol{x},y) \tag{2-34}$$

即希望 $\tilde{p}(\boldsymbol{x},y)=\tilde{p}(\boldsymbol{x})p(y|\boldsymbol{x})$，称 $p(\boldsymbol{x},y)=p(\boldsymbol{x})p(y|\boldsymbol{x})$为乘法准则。最大熵模型的约束希望在不同的特征函数 $f(\boldsymbol{x},y)$下通过估计 $p(y|\boldsymbol{x})$的参数来满足乘法准则。

由此，最大熵模型的学习过程可以转化为一个最优化问题的求解过程。即在给定若干特征提取函数

$$f_i(\boldsymbol{x},y),\quad i=1,2,\cdots,M \tag{2-35}$$

以及 y_i 的所有可能取值 $C=\{c_1,c_2,\cdots,c_K\}$的条件下，求解

$$\begin{aligned}\max\quad & H(p)=-\sum_{(\boldsymbol{x},y)\in D}\tilde{p}(\boldsymbol{x})p(y\mid\boldsymbol{x})\log p(y\mid\boldsymbol{x})\\ \text{s.t.}\quad & E_{\tilde{p}}(f_i)=E_p(f_i)\\ & \sum_{y\in C}p(y\mid\boldsymbol{x})=1\end{aligned} \tag{2-36}$$

将该最大化问题转化为最小化问题即 $\min -H(p)$，可用拉格朗日乘子法求解。拉格朗日函数为

$$\mathrm{Lag}(p,\boldsymbol{\omega})=-H(p)+\omega_0\Big(1-\sum_{y\in C}p(y\mid\boldsymbol{x})\Big)+\sum_{i=1}^{M}\omega_i(E_p(f_i)-E_{\tilde{p}}(f_i)) \tag{2-37}$$

其中，$\boldsymbol{\omega}=(\omega_0,\omega_1,\cdots,\omega_M)$为引入的拉格朗日乘子。通过最优化 $\mathrm{Lag}(p,\boldsymbol{\omega})$可求得

$$p_{\boldsymbol{\omega}}(y\mid\boldsymbol{x})=\frac{1}{Z_{\boldsymbol{\omega}}(\boldsymbol{x})}\exp\Big(\sum_{i=1}^{M}\omega_i f_i(\boldsymbol{x},y)\Big) \tag{2-38}$$

其中

$$Z_{\boldsymbol{\omega}}(\boldsymbol{x})=\sum_{y\in C}\exp\Big(\sum_{i=1}^{M}\omega_i f_i(\boldsymbol{x},y)\Big) \tag{2-39}$$

2.3.2　最大熵模型与逻辑回归之间的关系

分类问题中，假设特征函数个数 M 等于样本输入变量的个数 n，即 $n=M$。以二分类问题为例，定义如下特征函数，每个特征函数只提取一个属性的值

$$f_i(\boldsymbol{x},y)=\begin{cases}x_i & y=1\\ 0 & y=0\end{cases} \tag{2-40}$$

则

$$\begin{aligned}Z_{\boldsymbol{\omega}}(\boldsymbol{x})&=\sum_{y\in C}\exp\Big(\sum_{i=1}^{M}\omega_i f_i(\boldsymbol{x},y)\Big)\\&=\exp\Big(\sum_{i=1}^{M}\omega_i f_i(\boldsymbol{x},y=0)\Big)+\exp\Big(\sum_{i=1}^{M}\omega_i f_i(\boldsymbol{x},y=1)\Big)\\&=1+\exp(\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x})\end{aligned} \tag{2-41}$$

注意，此处 $\boldsymbol{\omega}=(\omega_1,\omega_2,\cdots,\omega_M)$，不包含 ω_0

$$\begin{cases}p(y=0\mid\boldsymbol{x})=\dfrac{1}{1+\mathrm{e}^{\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x}}}\\[2ex] p(y=1\mid\boldsymbol{x})=\dfrac{\mathrm{e}^{\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x}}}{1+\mathrm{e}^{\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x}}}\end{cases} \tag{2-42}$$

可以看到，此时最大熵模型等价于二分类逻辑回归模型。

对于多分类问题，可定义 $f_i(\boldsymbol{x},y=c_k)=\lambda_{ik}x_i$，则

$$p_{\boldsymbol{\omega}}(y=c_k\mid\boldsymbol{x})=\frac{1}{Z_{\boldsymbol{\omega}}(\boldsymbol{x})}\exp\Big(\sum_{i=1}^{M}\omega_i f_i(\boldsymbol{x},y)\Big)=\frac{1}{Z_{\boldsymbol{\omega}}(\boldsymbol{x})}\mathrm{e}^{\boldsymbol{\alpha}_k^{\mathrm{T}}\boldsymbol{x}} \tag{2-43}$$

其中

$$\begin{cases}Z_{\boldsymbol{\omega}}(\boldsymbol{x})=\displaystyle\sum_{y\in C}\exp\Big(\sum_{i=1}^{M}\omega_i f_i(\boldsymbol{x},y)\Big)=\sum_{k=1}^{K}\exp\Big(\sum_{i=1}^{M}\omega_i\lambda_{ik}x_i\Big)=\sum_{k=1}^{K}\mathrm{e}^{\boldsymbol{\alpha}_k^{\mathrm{T}}\boldsymbol{x}}\\ \boldsymbol{\alpha}_k^{\mathrm{T}}=(\omega_1\lambda_{1k},\omega_2\lambda_{2k},\cdots,\omega_M\lambda_{Mk})^{\mathrm{T}}\end{cases} \tag{2-44}$$

式(2-43)与式(2-24)等价，此时最大熵模型等价于多分类逻辑回归。最大熵模型可以通过拟牛顿法、梯度下降法等学习，本书略。

2.4　评价指标

对于一个分类任务，往往可以训练许多不同模型。那么，如何从众多模型中挑选出综合表现最好的那一个，这就涉及对模型的评价问题。接下来将介绍一些常用的模型评价指标。

2.4.1 混淆矩阵

混淆矩阵是理解大多数评价指标的基础，这里用一个经典表格来解释混淆矩阵是什么，如表 2-1 所示。

表 2-1 混淆矩阵示意表

真实值	预测值	
	0	1
0	True Negative(TN)	False Positive(FP)
1	False Negative(FN)	True Positive(TP)

显然，混淆矩阵包含四部分的信息：

(1) 真阴率(True Negative，TN)表明实际是负样本预测成负样本的样本数。

(2) 假阳率(False Positive，FP)表明实际是负样本预测成正样本的样本数。

(3) 假阴率(False Negative，FN)表明实际是正样本预测成负样本的样本数。

(4) 真阳率(True Positive，TP)表明实际是正样本预测成正样本的样本数。

对照混淆矩阵，很容易就能把关系、概念理清楚。但是久而久之，也很容易忘记概念。可以按照位置前后分为两部分记忆：前面的部分是 True/False 表示真假，即代表着预测的正确性；后面的部分是 Positive/Negative 表示正负样本，即代表着预测的结果。所以，混淆矩阵即可表示为正确性——预测结果的集合。现在再来看上述四个部分的概念：

(1) TN，预测是负样本，预测对了。

(6) FP，预测是正样本，预测错了。

(7) FN，预测是负样本，预测错了。

(8) TP，预测是正样本，预测对了。

大部分的评价指标都是建立在混淆矩阵基础上的，包括准确率、精确率、召回率、F1-score，当然也包括 AUC。

2.4.2 准确率

准确率是最为常见的一项指标，即预测正确的结果占总样本的百分比，其公式如下：

$$\text{Accuracy}=\frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}} \tag{2-45}$$

虽然准确率可以判断总的正确率，但是在样本不平衡的情况下，并不能作为很好的指标来衡量结果。假设在所有样本中，正样本占 90%，负样本占 10%，样本是严重不平衡的。模型将全部样本预测为正样本即可得到 90%的高准确率，如果仅使用准确率这单一指标进行评价，模型就可以像这样“偷懒”获得很高的评分。正因如此，也就衍生出了其他两种指标：精确率和召回率。

2.4.3 精确率与召回率

精确率(Precision)又叫查准率，它是针对预测结果而言的。精确率表示在所有被预测

为正的样本中实际为正的样本的概率。意思就是在预测为正样本的结果中，有多少把握可以预测正确，其公式如下：

$$\text{Precision}=\frac{\text{TP}}{\text{TP}+\text{FP}} \tag{2-46}$$

召回率(Recall)又叫查全率，它是针对原样本而言的。召回率表示在实际为正的样本中被预测为正样本的概率，其公式如下：

$$\text{Recall}=\frac{\text{TP}}{\text{TP}+\text{FN}} \tag{2-47}$$

召回率一般应用于漏检后果严重的场景下。例如在网贷违约率预测中，相比信誉良好的用户，我们更关心可能会发生违约的用户。如果模型过多地将可能发生违约的用户当成信誉良好的用户，后续可能会发生的违约金额会远超过好用户偿还的借贷利息金额，造成严重偿失。召回率越高，代表不良用户被预测出来的概率越高。

2.4.4　PR 曲线

分类模型对每个样本点都会输出一个置信度。通过设定置信度阈值，就可以完成分类。不同的置信度阈值对应着不同的精确率和召回率。一般来说，置信度阈值较低时，大量样本被预测为正例，所以召回率较高，而精确率较低；置信度阈值较高时，大量样本被预测为负例，所以召回率较低，而精确率较高。

PR 曲线就是以精确率为纵坐标，以召回率为横坐标做出的曲线，如图 2-2 所示。

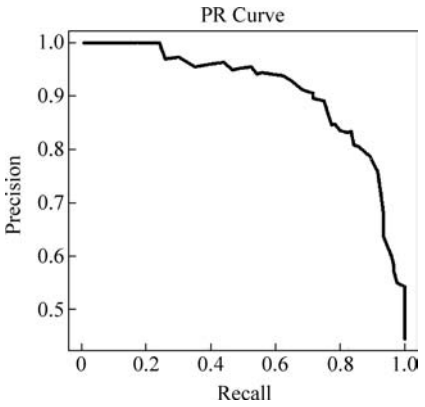

图 2-2　PR 曲线示意图

2.4.5　ROC 曲线与 AUC 曲线

对于某个二分类分类器来说，输出结果标签(0 还是 1)往往取决于置信度以及预定的置信度阈值。比如常见的阈值就是 0.5，大于 0.5 的认为是正样本，小于 0.5 的认为是负样本。如果增大这个阈值，预测错误(针对正样本而言，即指预测是正样本但是预测错误，下同)的概率就会降低，但是随之而来的就是预测正确的概率也降低；如果减小这个阈值，那么预测正确的概率会升高，但是同时预测错误的概率也会升高。实际上，这种阈值的选取一定程度上反映了分类器的分类能力。我们当然希望无论选取多大的阈值，分类都能尽可能地正确。为了形象地衡量这种分类能力，ROC 曲线进行了表征，如图 2-3 所示，为一条 ROC 曲线。

横轴：假阳率(False Positive Rate，FPR)

$$\text{FPR}=\frac{\text{FP}}{\text{TN}+\text{FP}} \tag{2-48}$$

纵轴：真阳率(True Positive Rate，TPR)

$$\text{TPR}=\frac{\text{TP}}{\text{TP}+\text{FN}} \tag{2-49}$$

显然，ROC 曲线的横纵坐标都在[0，1]之间，面积不大于 1。现在分析几个 ROC 曲线的特殊情况，更好地掌握其性质。

(0，0)：假阳率和真阳率都为 0，即分类器全部预测成负样本。

(0，1)：假阳率为 0，真阳率为 1，全部完美预测正确。

(1，0)：假阳率为 1，真阳率为 0，全部完美预测错误。

(1，1)：假阳率和真阳率都为 1，即分类器全部预测成正样本。

当 TPR＝FPR 为一条斜对角线时，表示预测为正样本的结果一半是对的，一半是错的，为随机分类器的预测效果。ROC 曲线在斜对角线以下，表示该分类器效果差于随机分类器；反之，效果好于随机分类器。当然，我们希望 ROC 曲线尽量位于斜对角线以上，也就是向左上角(0，1)凸。

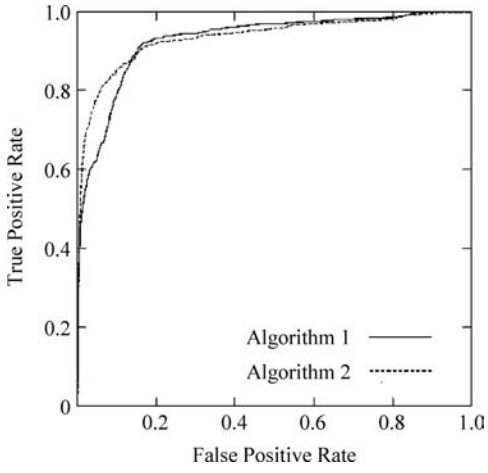

图 2-3　ROC 曲线示意图

2.5　实例：基于逻辑回归实现乳腺癌预测

本节基于乳腺癌数据集介绍逻辑回归的应用，模型的构造与训练如代码清单 2-1 所示。

代码清单 2-1　逻辑回归模型的构造与训练

```
from sklearn.datasets import load_breast_cancer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
cancer = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(
    cancer.data, cancer.target, test_size = 0.2)
model = LogisticRegression()
model.fit(X_train, y_train)
train_score = model.score(X_train, y_train)
test_score = model.score(X_test, y_test)
print('train score: {train_score:.6f}; test score: {test_score:.6f}'.format(
    train_score = train_score, test_score = test_score))
```

根据代码输出可知，模型在训练集上的准确率达到 0.969，在测试集上的准确率达到 0.921。为了进一步分析模型效果，代码清单 2-2 进一步评估了模型在测试集上的准确率、召回率以及精确率。三者分别达到了 0.921、0.960、0.923。

代码清单 2-2　模型评估

```
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
y_pred = model.predict(X_test)
accuracy_score_value = accuracy_score(y_test, y_pred)
recall_score_value = recall_score(y_test, y_pred)
precision_score_value = precision_score(y_test, y_pred)
classification_report_value = classification_report(y_test, y_pred)
print("准确率:", accuracy_score_value)
print("召回率:", recall_score_value)
print("精确率:", precision_score_value)
print(classification_report_value)
```

第 3 章

CHAPTER 3

k-近邻算法

k-近邻算法(k-Nearest Neighbor, KNN)是一种常用的分类或回归算法。给定一个训练样本集合 D 以及一个需要进行预测的样本 x,k-近邻算法的思想非常简单:对于分类问题,k-近邻算法从所有训练样本集合中找到与 x 最近的 k 个样本,然后通过投票法选择这 k 个样本中出现次数最多的类别作为 x 的预测结果;对于回归问题,k-近邻算法同样找到与 x 最近的 k 个样本,然后对这 k 个样本的标签求平均值,得到 x 的预测结果。k-近邻算法的描述如算法 3-1 所示。

算法 3-1　k-近邻算法

输入:训练集 $D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_m,y_m)\}$;$k$ 值;待预测样本 x;如果是 k-近邻分类,同时给出类别集合 $C=\{c_1,c_2,\cdots,c_K\}$

输出:样本 x 所属的类别或预测值 y

1. 计算 x 与所有训练集合中所有样本之间的距离,并从小到大排序,返回排序后样本的索引.

$$P=\underset{i}{\operatorname{argsort}}\{d(\boldsymbol{x},\boldsymbol{x}_i)\mid i=1,2,\cdots,m\}$$

2. 对于分类问题,投票挑选出前 k 个样本中包含数量最多的类别

$$\text{Return } y=\underset{i=1,2,\cdots,K}{\arg\max}\sum_{p\in P}I(\boldsymbol{x}_p=c_i)$$

3. 对于回归问题,用前 k 个样本标签的均值作为 x 的估计值

$$\text{Return } y=\frac{1}{k}\sum_{p\in P}y_p$$

对 k-近邻算法的研究包含三个方面:k 值的选取、距离的度量和如何快速地进行 k 个近邻的检索。

3.1　k 值的选取

投票法的准则是少数服从多数,所以当 k 值很小时,得到的结果就容易产生偏差。最近邻算法是这种情况下的极端,也就是 $k=1$ 时的 k-近邻算法。最近邻算法中,样本 x 的预测结果只由训练集中与其距离最近的那个样本决定。

如果 k 值选取较大,则可能会将大量其他类别的样本包含进来,极端情况下,将整个训练集的所有样本都包含进来,这样同样可能会造成预测错误。一般情况下,可通过交叉验

证、在验证集上多次尝试不同的 k 值来挑选最佳的 k 值。

3.2 距离的度量

对于连续变量，一般使用欧氏距离直接进行距离度量。对于离散变量，可以先将离散变量连续化，然后再使用欧氏距离进行度量。

词嵌入(Word Embedding)是自然语言处理领域常用的一种对单词进行编码的方式。词嵌入首先将离散变量进行热独(one-hot)编码，假定共有 5 个单词{A,B,C,D,E}，则对 A 的热独编码为$(1,0,0,0,0)^{\mathrm{T}}$，B 的热独编码为$(0,1,0,0,0)^{\mathrm{T}}$，其他单词类似。编码后的单词用矩阵表示为

$$\boldsymbol{X}=\begin{array}{c} \begin{matrix} A & B & C & D & E \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \tag{3-1}$$

随机初始化一个用于词嵌入转化的矩阵 $\boldsymbol{M}_{d\times 5}$，其中每一个 d 维的向量表示一个单词。词嵌入后的单词用矩阵表示为

$$\boldsymbol{E}=\boldsymbol{M}_{d\times 5}\boldsymbol{X}=\begin{array}{c} \begin{matrix} A & B & C & D & E \end{matrix} \\ \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{d1} & x_{d2} & x_{d3} & x_{d4} & x_{d5} \end{pmatrix} \end{array} \tag{3-2}$$

矩阵 $\boldsymbol{E}$ 中的每一列是相应单词的词嵌入表示，d 是一个超参数，$\boldsymbol{M}$ 可以通过深度神经网络(见第 11 章)在其他任务上进行学习，之后就能用单词词嵌入后的向量表示计算内积，用以表示单词之间的相似度。对于一般的离散变量同样可以采用类似词嵌入的方法进行距离度量。

3.3 快速检索

当训练集合的规模很大时，如何快速找到样本 x 的 k 个近邻成为计算机实现 k-近邻算法的关键。一个朴素的思想是：

(1) 计算样本 x 与训练集中所有样本的距离。

(2) 将这些点依据距离从小到大进行排序选择前 k 个。

算法的时间复杂度是计算到训练集中所有样本距离的时间加上排序的时间。该算法的第(2)步可以用数据结构中的查找序列中前 k 个最小数的算法优化，而不必对所有距离都进行排序。一个更为可取的方法是为训练样本事先建立索引，以减少计算的规模。kd 树是一种典型的存储 k 维空间数据的数据结构(此处的 k 指 x 的维度大小，与 k-近邻算法中的 k

没有任何关系)。建立好 kd 树后,给定新样本后就可以在树上进行检索,这样就能够大大降低检索 k 个近邻的时间,特别是当训练集的样本数远大于样本的维度时。关于 kd 树的详细介绍可参考文献[3]。

3.4 实例:基于 k-近邻算法实现鸢尾花分类

本节以鸢尾花(Iris)数据集的分类来直观理解 k-近邻算法。为了在二维平面展示鸢尾花数据集,这里使用花萼宽度和花瓣宽度两个特征进行可视化,如图 3-1 所示。

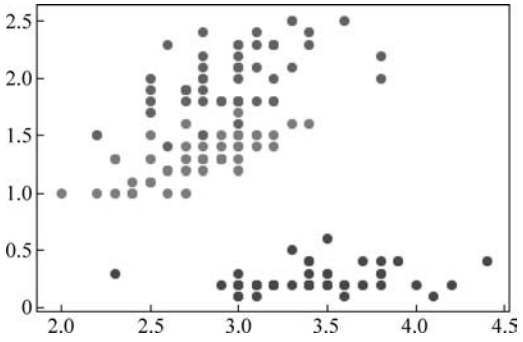

图 3-1 鸢尾花数据集(见彩插)

图中蓝色数据点表示 Setosa,橙色数据点表示 Versicolour,绿色数据点表示 Virginica。sklearn 中提供的 k-近邻模型称为 KNeighborsClassifier,代码清单 3-1 给出了模型构造和训练代码。

代码清单 3-1 k-近邻模型的构造与训练

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier as KNN
if __name__ == '__main__':
    iris = load_iris()
    x_train, x_test, y_train, y_test = train_test_split(
        iris.data[:, [1,3]], iris.target)
    model = KNN()
    model.fit(x_train, y_train)
```

代码清单 3-2 对上述模型进行了测试。根据程序输出可以看出,模型在训练集上的准确率达到 0.964,测试集上的准确率达到 0.947。

代码清单 3-2 模型测试

```
train_score = model.score(x_train, y_train)
test_score = model.score(x_test, y_test)
print("train score:", train_score)
print("test score:", test_score)
```