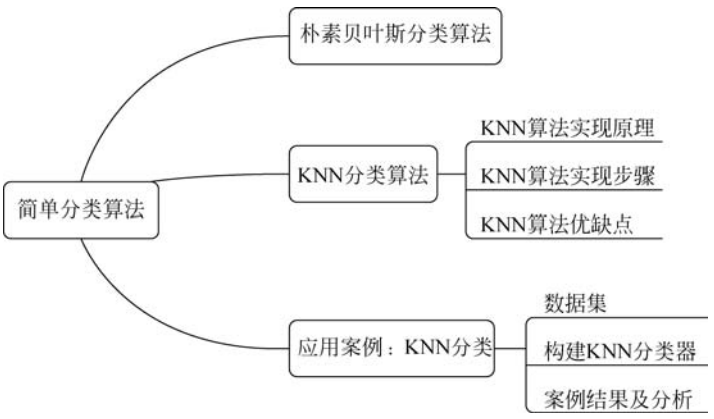
第3章

简单分类算法

[思维导图]



前两章介绍了机器学习基础知识以及数据预处理方法，本章将讲解机器学习中最常见的一种任务——分类任务。

分类是数据挖掘、机器学习和模式识别中一个重要的研究领域。数据分类研究就是给数据“贴标签”进行分类，类别分得越精准，得到的结果就越有价值。分类是一个有监督的学习过程，目标数据库中有哪些类别是已知的，分类过程需要做的就是把每一条记录都归到对应的类别之中。分类前必须事先知道各个类别的信息，并且所有待分类的数据条目都默认有对应的类别。分类的大致流程为：通过已有数据集（也称训练集）的学习，训练得到一个目标函数 f（也称模型），把每个数据集的元素 x 映射到目标类别 y，且 y 必须是离散的。

分类算法按模型可以分为两类：预测性模型与概率性模型。其中，预测性模型可以直接判定数据应被分到哪个类，而概率性模型不会直接得到结果，只能确定属于每个类的概率。分类算法按照原理又可以分为以下四大类。

(1) 基于统计的分类，如朴素贝叶斯分类算法。

(2) 基于规则的分类，如决策树算法。

(3) 基于神经网络的分类，如神经网络算法。

(4) 基于距离的分类，如 KNN 分类算法。

本章主要介绍朴素贝叶斯分类算法和 KNN 分类算法，决策树算法和神经网络算法在本书其他章节详细介绍。

3.1 朴素贝叶斯分类算法

朴素贝叶斯(Naive Bayes)分类算法是利用概率统计进行分类的算法。该算法基于贝叶斯定理，其基本思想是：对于一个待分类项，求解属于各个类别的概率，概率最大的类别即为待分类项的类别。

设 x 为一个待分类项，C 为一个类别集合，c_i 是第 i 个类别。根据贝叶斯定理，x 属于 c_i 的概率如式(3.1)所示。

$$P(c_i \mid x)=\frac{P(x \mid c_i)P(c_i)}{P(x)} \tag{3.1}$$

其中，$P(x|c_i)$为类别条件概率；$P(c_i)$为 c_i 的先验概率，一般通过收集的数据集统计计算得到。$P(x)$用于归一化，对于给定的样本数据集，$P(x)$对于所有类别是相同的，因此对于一般的分类问题通常不计算 $P(x)$。

朴素贝叶斯分类算法的目标就是选择 $P(c_i|x)$最大的 c_i 作为 x 所属的类别，求解 $P(c_i|x)$的关键在于计算类别条件概率 $P(x|c_i)$。下面介绍 $P(x|c_i)$的计算过程。

设 a_k 是 x 的第 k 个特征属性，朴素贝叶斯分类算法假设这些特征属性相互独立，$P(x|c_i)$的计算如式(3.2)所示。

$$P(x \mid c_i)=P(a_0 \mid c_i)P(a_1 \mid c_i)\cdots P(a_n \mid c_i)=\prod_{k=0}^{n}P(a_k \mid c_i) \tag{3.2}$$

将式(3.2)代入式(3.1)，$P(c_i|x)$可推导得式(3.3)。

$$P(c_i \mid x)=\frac{P(c_i)}{P(x)}\prod_{k=0}^{n}P(a_k \mid c_i) \tag{3.3}$$

由于 $P(x)$对于所有类别是相同的，因此 $P(c_i|x)$与式(3.3)中分子成正比，可写作式(3.4)。

$$P(c_i \mid x) \propto P(c_i)\prod_{k=0}^{n}P(a_k \mid c_i) \tag{3.4}$$

最终朴素贝叶斯的判定公式如式(3.5)所示。

$$H(x)=\arg\max P(c_i)\prod_{k=0}^{n}P(a_k \mid c_i) \tag{3.5}$$

下面给出一个购买商品的例子，说明如何利用朴素贝叶斯分类算法进行分类。假设顾客根据外观、质量和价格三个特征属性决定是否购买某一商品，表 3-1 展示了已收集的商品购买样本数据。

表 3-1　已收集的商品购买样本数据

外　观	质　量	价　格	是否购买
好	好	适中	是
不好	好	低	否
不好	不好	高	否
好	好	低	是
不好	好	适中	是
好	不好	低	否
好	好	高	是
不好	不好	适中	否
好	不好	适中	是
不好	好	高	否

现有商品 x，特征属性为外观好、质量不好、价格高，需要预测顾客是否购买该商品。利用朴素贝叶斯分类算法完成该任务的具体步骤如下。

(1) 统计样本数据中“买”类别的先验概率。

$$P(\text{买})=\frac{5}{10}=\frac{1}{2}$$

(2) 统计样本数据中“不买”类别的先验概率：

$$P(\text{不买})=\frac{5}{10}=\frac{1}{2}$$

(3) 统计样本数据中“买”类别下，商品 x 拥有的特征属性的条件概率：

$$P(\text{外观好}\mid\text{买})=\frac{4}{5}$$

$$P(\text{质量不好}\mid\text{买})=\frac{1}{5}$$

$$P(\text{价格高}\mid\text{买})=\frac{1}{5}$$

(4) 计算购买商品 x 的概率：

$$P(\text{买}\mid x)=\frac{1}{2}\times\frac{4}{5}\times\frac{1}{5}\times\frac{1}{5}=\frac{2}{125}$$

(5) 计算样本数据中“不买”类别下，商品 x 拥有的特征属性的条件概率：

$$P(\text{外观好}\mid\text{不买})=\frac{1}{5}$$

$$P(\text{质量不好}\mid\text{不买})=\frac{3}{5}$$

$$P(\text{价格高}\mid\text{不买})=\frac{2}{5}$$

(6) 计算不买商品 x 的概率：

$$P(\text{不买}\mid x)=\frac{1}{2}\times\frac{1}{5}\times\frac{3}{5}\times\frac{2}{5}=\frac{3}{125}$$

由于 $P(\text{不买}|x)>P(\text{买}|x)$，预测结果为不购买商品 x。

算法 3-1　朴素贝叶斯分类算法

输入：样本数据集 D，类别集合 $C=\{c_1,c_2,\cdots,c_m\}$，特征属性集合 $A=\{a_0,a_1,\cdots,a_n\}$，待预测样本 x

输出：待预测样本 x 所属类别

(1) 根据 D 中数据，统计每个类别的先验概率 $P(c_i)$；

(2) 根据 D 中数据，统计每个类别下待预测样本 x 所有特征属性的条件概率 $P(a_k|c_i)$；

(3) 根据式(3.4)，计算待预测样本 x 属于每个类别的概率 $P(c_i|x)$；

(4) 根据式(3.5)，取 $P(c_i|x)$值最大的类别作为待预测样本 x 所属的类别。

算法 3-1 描述了朴素贝叶斯分类算法的步骤。

朴素贝叶斯分类算法效率稳定，预测过程简单快速，对于多分类问题同样很有效，常用于文本分类、垃圾文本过滤、情感判别、多分类实时预测等任务。但朴素贝叶斯分类算法假设所有特征属性对分类的影响是相互独立的，当这些特征属性存在关联性时分类效果会变差。

3.2　KNN 分类算法

KNN(K-Nearest Neighbor，K 近邻)分类算法是机器学习分类技术中最简单的算法之一，其指导思想是“近朱者赤，近墨者黑”，即由你的邻居来推断出你的类别。

3.2.1　KNN 算法实现原理

为了判断未知样本的类别，以所有已知类别的样本作为参照，计算未知样本与所有已知样本的距离，从中选取与未知样本距离最近的 K 个已知样本，根据少数服从多数的投票法则(Majority-Voting)，将未知样本与 K 个最近邻样本中所属类别占比较多的归为一类。以上即为 KNN 算法在分类任务中的基本原理。其中，K 表示要选取的最近邻样本实例的个数，可以根据实际情况进行选择。在 sklearn 库中 KNN 算法的 K 值是通过 n_neighbors 参数来调节的，默认值为 5。

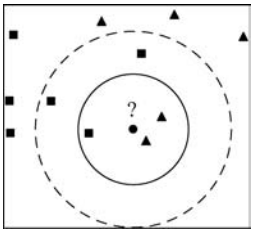


图 3-1　KNN 分类示意图

图 3-1 展示了 KNN 分类示意图。如何判断圆形应该属于哪一类？是属于三角形还是属于四边形？如果 $K=3$，由于三角形所占比例为 2/3，圆形将被判定为属于三角形类；如果 $K=5$，由于四边形比例为 3/5，因此圆形将被判定

为属于四边形类。

由于 KNN 分类算法在分类决策时只依据最近邻的一个或者几个样本的类别来决定待分类样本所属的类别，而不是靠判别类域的方法来确定所属的类别，因此对于类域的交叉或重叠较多的待分样本集来说，KNN 算法较其他算法更为适合。

3.2.2　KNN 算法实现步骤

KNN 算法的实现分如下四步。

(1) 样本特征量化。

样本的所有特征都要做可比较的量化，若样本特征中存在非数值类型，则必须采取手段将其量化为数值。例如，样本特征中包含颜色，可通过将颜色转换为灰度值来实现距离计算。

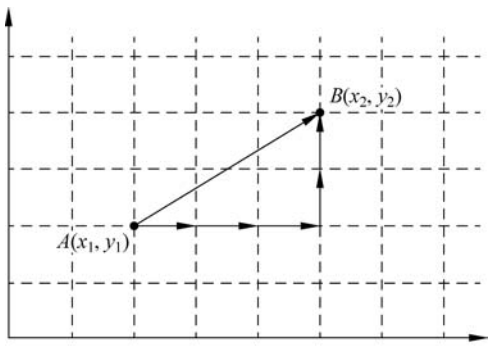
(2) 样本特征归一化。

样本有多个参数，每一个参数都有自己的定义域和取值范围，它们对距离计算的影响不一样，如取值较大的影响力会盖过取值较小的参数。所以，对样本参数必须做一些比例处理，最简单的方式即对所有特征的数值都采取归一化处置。

(3) 计算样本之间的距离。

需要一个距离函数以计算两个样本之间的距离，通常使用的距离函数有欧几里得距离(简称欧氏距离)、余弦距离、汉明距离和曼哈顿距离等，一般选欧氏距离作为距离度量，但是这些只适用于连续变量。在文本分类这种非连续变量情况下，汉明距离可以用来作为度量。通常情况下，如果运用一些特殊的算法来计算度量，K 近邻分类准确率可显著提高，如运用大边缘最近邻法或者近邻成分分析法。

图 3-2 展示了欧氏距离与曼哈顿距离。以二维空间中的 $A(x_1,y_1)$、$B(x_2,y_2)$两点为例，分别用欧氏距离与曼哈顿距离度量两点之间的距离。



欧氏距离

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

曼哈顿距离

$$\text{ManhattanDistance}(d)=|x_2-x_1|+|y_2-y_1|$$

图 3-2　欧氏距离与曼哈顿距离

(4) 确定 K 值。

K 值选得太大易引起欠拟合,太小容易过拟合,需交叉验证确定 K 值。

3.2.3 KNN 算法优缺点

KNN 算法的优点包括:

(1) 简单,易于理解,易于实现,无须估计参数,无须训练;

(2) 适合对稀有事件进行分类;

(3) 特别适合于多分类问题(Multi-label,对象具有多个类别标签)。

KNN 算法在分类问题上的不足之处在于:当样本不平衡时,即一个类的样本数量很大,而其他类样本数量很小时,有可能导致当输入一个新样本时,该样本的 K 个邻居中大数量类的样本容易占多数,导致错误分类。因此,KNN 算法可以采用加权算法的方法来改进。比如,对样本距离小的邻域数据赋予更大的权值。

KNN 算法的主要使用场景包括文本分类、用户推荐等。

3.3 应用案例:KNN 分类

本节以一个实际分类例子阐述 KNN 分类过程。

3.3.1 数据集

第 2 章使用鸢尾花数据集,基于 PCA 进行数据降维,将鸢尾花数据的四维特征降至二维,目的是更好地可视化数据特征。本章介绍利用 KNN 算法对原始鸢尾花数据集(四维特征)进行分类。

3.3.2 构建 KNN 分类器

首先获取鸢尾花数据集,然后通过绘制散点图,初步判断鸢尾花数据是否适合 KNN 分类,代码如例 3-1 所示。

【例 3-1】 绘制鸢尾花数据集散点图。

```
import pandas as pd
import mglearn
#导入数据集
iris_dataframe = pd.DataFrame(X_train,columns = iris_dataset.feature_names)
#按 y_train 着色
grr = pd.plotting.scatter_matrix(iris_dataframe,c = y_train,figsize = (15,15),marker = 'o',
hist_kwds = {'bins':20},s = 60,alpha = .8,cmap = mglearn.cm3)
```

图 3-3 展示了鸢尾花数据散点分布。3 种鸢尾花散点已经按照分类进行着色。可以很清晰地看到,现有数据基本上可以将 3 种花分类,各个颜色的散点基本可以形成群落。

下一步通过 sklearn 库使用 Python 构建一个 KNN 分类模型,主要包括如下步骤:

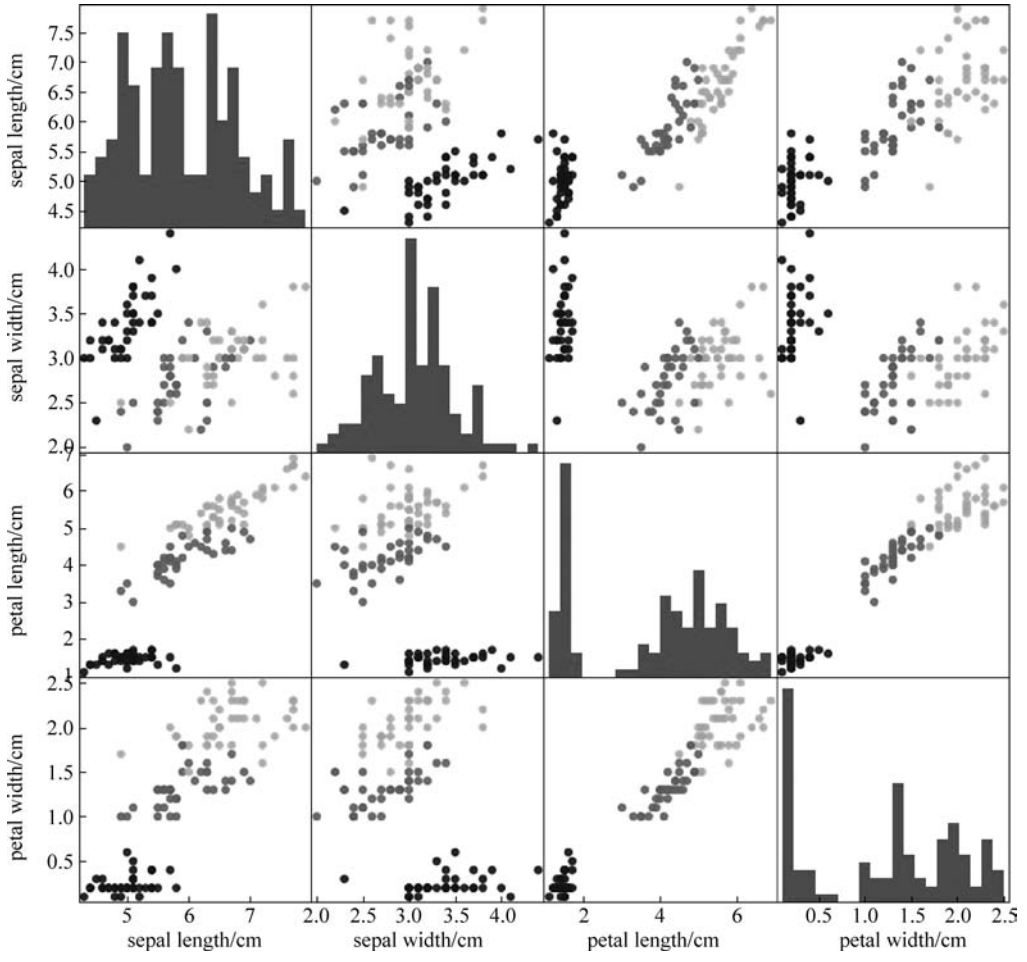


图 3-3　鸢尾花数据散点分布

（1）初始化分类器参数（只有少量参数需要指定，其余参数保持默认即可）；

（2）训练模型；

（3）评估、预测。

KNN 算法的 K 是指几个最近邻居，这里构建一个 $K=3$ 的模型，并且将训练数据 X_train 和 Y_train 作为参数，构建模型的代码如例 3-2 所示。

【例 3-2】 构建 KNN 模型。

```
1.  from sklearn.neighbors import KNeighborsClassifier
2.  # 调用 sklearn 库中的 KNN 模型
3.  knn = KNeighborsClassifier(n_neighbors = 3)
4.  knn.fit(X_train, y_train)
```

注意，KNN 是一个对象，knn.fit()函数实际上修改的是 KNN 对象的内部数据。现在 KNN 分类器已经构建完成，使用 knn.predict()函数可以对数据进行预测，为了评估分

类器的准确率，将预测结果与测试数据进行对比，计算分类准确率。

3.3.3 案例结果及分析

调用例 3-2 中构建的 KNN 模型进行预测，输出预测结果并计算准确率。代码如例 3-3 所示。

【例 3-3】 预测结果计算准确率。

```
y_pred = knn.predict(X_test)
print("Test set predictions:\n {}".format(y_pred))
print("Test set score: {:.2f}".format(np.mean(y_pred = = y_test)))
```

输出结果：

```
Test set predictions:
  [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 2 1 0 2 2 1 0 2]
Test set score: 0.97
```

从结果可知，KNN 分类准确率可以达到 97%。