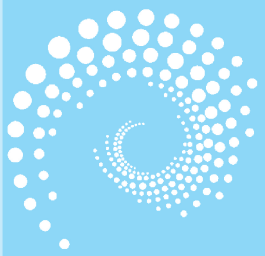
# 第5章

# K近邻

CHAPTER 5



**本章学习目标**

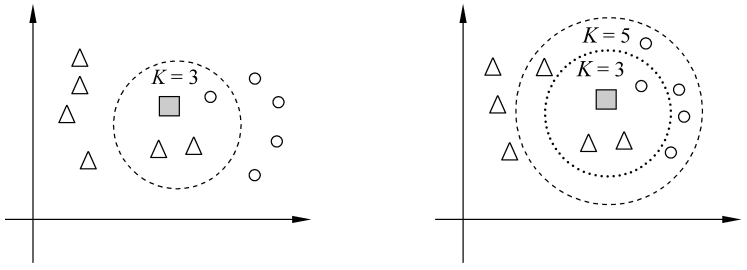
- 认知类目标：熟悉K近邻算法的基本思想。
- 价值类目标：理解K近邻算法及常用距离度量。
- 方法类目标：使用K近邻算法解决相关问题。
- 情感、态度、价值观类目标：了解K近邻算法在相关学科领域和社会实践中应用，了解最新发展动态以及相关发展状况，能灵活运用其解决实际问题，培养学生理论结合实践的科学态度、创新意识和社会责任感，激发学生为我国人工智能发展的潜在服务意识。

本章主要介绍K近邻算法的基本思想，K近邻算法中度量数据点之间距离(或相似性)的常用方法，以及K选择的重要性和K选择的正确方法。以鸢尾花数据集为例介绍K近邻算法的分类方法。在知识扩展部分，介绍几种数据点距离度量的方法。

## 5.1　K 近邻算法的基本思想

K 近邻算法也称为 KNN 算法，是最简单、最常用的分类算法。K 近邻算法在预测某个数据点的类别时，参考周围距离最近的 $K$ 个邻居的类别标签来决定该数据的类别。具体方法为，根据一个测试数据点，计算它与数据集中其他数据点的距离；找出距离最近的 $K$ 个数据点，作为该数据点的近邻数据点集合；根据这 $K$ 个近邻数据点所属的类别，来决定当前数据点的类别。

例如，在图 5.1 中，三角形表示类别一的数据点，圆形表示类别为二的数据点，现在确定灰色方块的类别。图中的圆圈表示 $K$ 个最近的邻居所在的区域。在 $K=3$ 的圆圈里面，类别为一的数据点有 2 个，类别为二的数据点有 1 个。采用少数服从多数的原则，灰色数据点的分类确定为类别一。在 $K=5$ 的圆圈里面，类别为一的数据点 3 个，类别为二的数据点为 5 个，由此可以判定灰色数据点的类别为二。




(a) $K$=3，目标点类别是类别一　　(b) $K$=5，目标点类别是类别二

图 5.1　KNN 算法实例

在 K 近邻算法中有两个重要问题：①数据点的距离的计算方法；②$K$ 的选择。下面将介绍这两个问题的解决方法。

## 5.2　K 近邻算法中的距离度量

K 近邻算法中，度量数据点之间的距离是一个非常重要的步骤。度量距离的常见方法有欧氏距离、余弦相似度、曼哈顿距离等。

### 1. 欧氏距离

欧氏距离就是 $n$ 维欧氏空间中亮点之间的距离。对于$\mathbb{R}^n$ 空间中两个点 $x$ 和 $y$，它们之间的距离定义为：

$$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2} \tag{5.1}$$

在使用欧氏距离时应将特征向量的每个分量归一化，以减少特征值的尺度范围不同所带来的干扰，避免数值小的特征分量会被数值大的特征分量淹没。

欧氏距离的优点是方法简单，缺点是计算量大，不适用于高维数据的情况，同时也没有考虑特征向量的概率分布。

### 2. 余弦相似度

对于高维数据,在运算中只考虑向量的方向而不考虑向量的大小时,使用余弦相似度来度量数据点之间的距离。余弦相似度就是两个向量夹角的余弦。两个方向完全相同的向量的余弦相似性为1,而两个完全相反的向量的相似性为-1。注意大小并不重要,因为这是方向的量度。

$$D(x,y)=\cos(\theta)=\frac{x\cdot y}{\|x\|\ \|y\|}$$

余弦相似度的主要问题是没有考虑向量的大小,只考虑了向量的方向。在实践中,这意味着没有充分考虑值的差异。余弦相似性经常用于文本分类过程中,当数据以单词计数表示时,经常使用此度量。

### 3. 曼哈顿距离

曼哈顿距离是一种概率意义上的距离,给定两个(向量)点 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 以及矩阵 $\boldsymbol{S}$,则曼哈顿距离定义为:

$$d(\boldsymbol{x},\boldsymbol{y})=\sqrt{(\boldsymbol{x}-\boldsymbol{y})^{\mathrm{T}}\boldsymbol{S}(\boldsymbol{x}-\boldsymbol{y})} \tag{5.2}$$

其中,要求矩阵 $\boldsymbol{S}$ 必须为正定的。这种距离度量的是两个随机向量的相似度。当矩阵 $\boldsymbol{S}$ 为单位矩阵 $\boldsymbol{I}$ 时,曼哈顿距离就退化成了欧氏距离。矩阵 $\boldsymbol{S}$ 可以通过计算训练样本集的协方差矩阵得到,也可以通过训练样本学习得到。

曼哈顿距离与欧氏距离相比更不直观,尤其是在高维数据的使用中。当数据集具有离散或二进制属性时,则倾向于使用曼哈顿距离,因为它考虑了在这些属性值中实际可以采用的路径,而欧氏距离的可行性较差。

## 5.3　选择合适的 $K$ 值

通过图5.1可知 $K$ 的取值比较重要,那么该如何确定 $K$ 取多少值好呢?一般可以通过交叉验证方法,选取一个较小的 $K$ 值并不断增加,然后计算验证集合的方差,从而确定一个比较合适的 $K$ 值。

通过交叉验证方法计算不同 $K$ 值的方差,一般会得到图5.2。

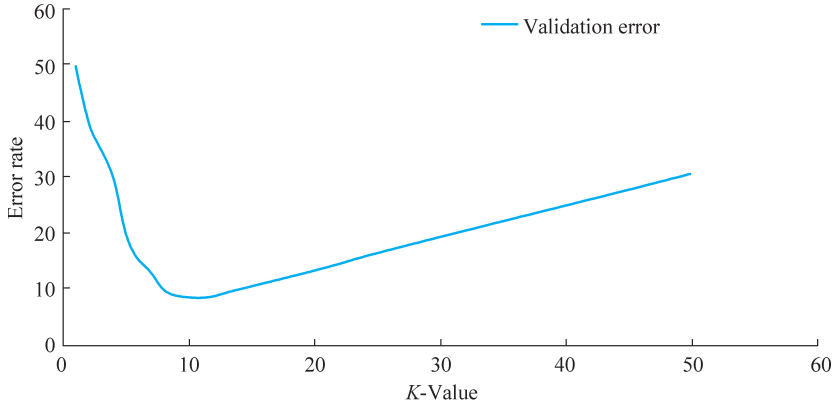



图 5.2　交叉验证方法寻找合适 $K$ 值

选择合适 $K$ 值时，需要一个临界点 $K_0$，当 $K$ 继续增大或减小的时候，错误率都会上升，如图 5.2 中 $K$ 的临界值是 10，所以选择 $K=10$ 比较合适。

## 5.4　K 近邻案例——鸢尾花分类

### 1. 案例说明

在本案例中，我们使用 iris 数据集。该数据集是机器学习的经典数据集，如表 5.1 所示。

表 5.1　iris 数据集格式

| 列　　号 | 列　　名 | 具 体 含 义 |
|---|---|---|
| 1 | sepal-width | 花萼宽度，数值数据 |
| 2 | sepal-length | 花萼长度，数值数据 |
| 3 | petal-width | 花瓣宽度，数值数据 |
| 4 | petal-length | 花瓣长度，数值数据 |
| 5 | species | 花的种类，类别变量：Setosa、Versicolour、Virginica |

### 2. 核心代码

下面展示采用 K 近邻算法实现鸢尾花数据集分类预测的核心代码。

（1）导入包。

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

（2）加载数据集。

```
dataset = pd.read_csv("./data/iris.csv")
dataset.head()
dataset['species'] = dataset['species'].map({'setosa': 0, 'versicolor': 1, 'virginica': 2})
# 查看数据集的基本性质
dataset.info()
```

（3）构建训练集和测试集。

```
X = dataset.iloc[:,:-1].values
y = dataset.iloc[:,4].values
# 构建训练集和测试集
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20)
```

(4) 构建模型。

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5)
classifier.fit(X_train,y_train)
y_pred = classifier.predict(X_test)
```

(5) 评估模型性能。

```
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

### 3. 结果展示

运行结果如图 5.3 和图 5.4 所示。

```
[[ 9  0  0]
 [ 0  9  0]
 [ 0  1 11]]
```

图 5.3　KNN 模型的混淆矩阵

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 9 |
| 1 | 0.90 | 1.00 | 0.95 | 9 |
| 2 | 1.00 | 0.92 | 0.96 | 12 |
| accuracy | | | 0.97 | 30 |
| macro avg | 0.97 | 0.97 | 0.97 | 30 |
| weighted avg | 0.97 | 0.97 | 0.97 | 30 |

图 5.4　KNN 模型的分类性能

选择合适的 $K$ 值。

```
error = []
for i in range(1,40):
    knn = KNeighborsClassifier(n_neighbors = i)
    knn.fit(X_train,y_train.astype('int'))
    pred_i = knn.predict(X_test)
    error.append(np.mean(pred_i != y_test.astype('int')))
plt.figure(figsize = (8,5))
plt.plot(range(1,40),error,color = "red",linestyle = "dashed",
         marker = "o",markerfacecolor = "blue",markersize = 10)
plt.title("Error Rate K Value")
plt.xlabel("K Value")
plt.ylabel("Mean Error")
```

误分率和 $K$ 值的关系图如图 5.5 所示。当 $K$ 取值在[13,15]的时候,分类器的错误率最低,故取这个范围内的任何一个值都可以。

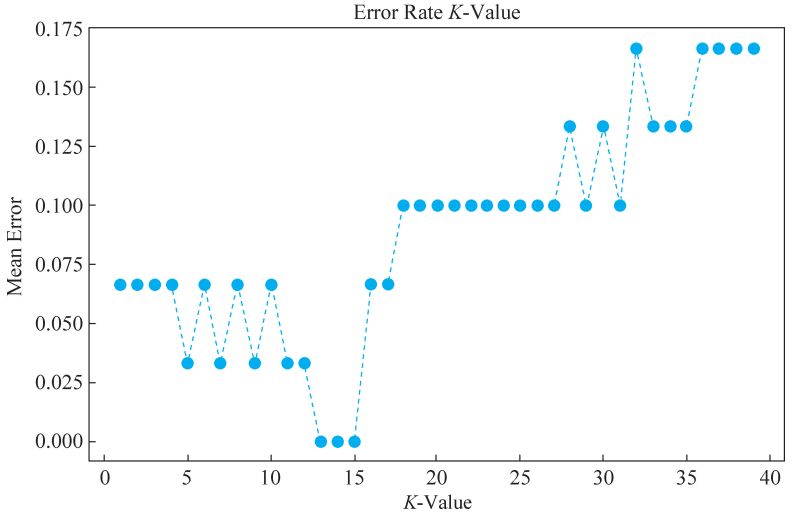



图 5.5　误分率和 K 值的关系图

## 5.5　知识扩展

### 1. 汉明距离

汉明距离是两个向量之间不同值的个数。它通常用于比较两个相同长度的二进制字符串。它还可以用于字符串，通过计算不同字符的数量来比较它们之间的相似程度。汉明距离的特点是要求所有向量的长度必须相等，当两个向量的长度不相等时，很难使用汉明距离。

### 2. 切比雪夫距离

切比雪夫距离定义为两个向量在任意坐标维度上的最大差值。换句话说，它就是沿着一个轴的最大距离，它通常被称为棋盘距离。

$$D(x,y)=\max_i(|x_i-y_i|)$$

切比雪夫距离很难像欧氏距离或余弦相似度那样用作通用的距离度量，通常用于非常特定的用例。因此，建议只在绝对确定它适合你的实例时才使用它。在实践中，切比雪夫距离经常用于仓库物流，或者在允许无限制的 8 向移动的游戏中使用。

### 3. 闵可夫斯基距离

闵可夫斯基距离比大多数测量都要复杂一些。它是一个在赋范向量空间（n 维实空间）中使用的度量。使用闵可夫斯基距离的赋范向量空间的要求如下。

- 0 向量：0 向量的长度是 0，而其他向量的长度都是正的。
- 标量因子：当你用一个正数乘以向量时，它的长度会改变，同时保持方向不变。
- 满足三角形不等式：两点之间最短的距离是一条直线。

闵可夫斯基距离公式如下：

$$D(x,y)=\left(\sum_{i=1}^{n}|x_i-y_i|^p\right)^{\frac{1}{p}}$$

闵可夫斯基距离需要关注的是参数 $p$ 的使用。我们可以通过设定不同的 $p$ 值来操作距离度量，使其与其他度量非常相似。若 $p=1$，则闵可夫斯基距离演化为曼哈顿距离；若 $p=2$，则为欧氏距离；若 $p=\infty$，则为切比雪夫距离。

闵可夫斯基距离理解的基础是曼哈顿距离、欧几里得距离和切比雪夫距离。在使用闵可夫斯基距离时，选择合适的参数 $p$ 是件非常麻烦的事情，需要根据实际的数据集查找恰当的 $p$ 值。

#### 4. Jaccard 索引

Jaccard 索引是一个用于计算样本集的相似性和多样性的度量，它是集合之间相似实体的总数除以实体的总数，即样本集交集的大小除以样本集并集的大小。例如，如果两个集合有一个共同的实体，而总共有 5 个不同的实体，那么 Jaccard 索引将是 1/5=0.2。

为了计算 Jaccard 距离，我们只需从 1 中减去 Jaccard 索引：

$$D(x,y)=1-\frac{|x\cap y|}{|y\cup x|}$$

Jaccard 索引的一个主要缺点是它受数据大小的影响很大。大型数据集可能对索引有很大的影响，因为它可以显著增加并集，同时保持交集相似。Jaccard 索引经常用于使用二进制或二进制化数据的应用程序中。当一个深度学习模型预测一幅图像(例如一辆汽车)的片段时，Jaccard 索引就可以用来计算真实标签的预测片段的准确性。同样，它也可以用于文本相似度分析，以衡量文档之间的选词重叠程度。

## 5.6　习题

1. 简述 K 近邻算法的基本思想。
2. 简述欧氏距离、曼哈顿距离以及余弦相似度的计算方法和适用场合。
3. 编写代码实现 $K$ 值的计算和选择。