第五讲 分类算法 (Classification)

2025/04/12

文件：机器学习实践课2.pdf、PJ-classification.zip
作业：机器学习-分类

1. 分类算法

1.1 二分类与多分类

- 二分类：分为两类
- 多分类：分为多类

2. KNN 分类算法

- **K 最近邻 (KNN, K Nearest Neighbor) 分类算法**是一种**监督学习**算法，是机器学习分类方法中最简单的方法之一
- KNN 中的 K ，指的是 K 最近邻，即选择与测试数据最接近的 K 个点来预测测试数据的类别
- 度量空间中点距离，有好几种度量方式，比如常见的**曼哈顿距离**、**欧氏距离**等
- KNN 算法常用的是**欧氏距离**，在二维平面中即两点之间的直线距离，计算公式如下：

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

- 若要拓展到多维空间，则公式变为：

$$d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n + y_n)^2} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

2.1 KNN 分类算法包括以下 4 个步骤：

1. 准备数据，对数据进行预处理
2. 计算测试样本点（也就是待分类点）到训练样本的距离
3. 对每个距离进行排序，然后选出距离最小的 K 个点
4. 对 K 个点所属的类别进行比较，根据少数服从多数的原则，将测试样本点归入在 K 个点中占比最高的那一类

2.2 性能分析：

- 优点：
  - 实现简单
  - 处理大数据时表现良好
  - 形成的决策边界可以是任意形状的
- 缺点：
  - 维数灾难，且可能被无关属性影响距离
  - K 值对结果影响大
  - 数据量大，计算量大

3. SVM 分类算法

3.1 感知机

- **感知机**时二分类的线性模型，其输入是实例的特征向量，输出的是实例的类别，分别是 $+1$ 和 -1 ，属于判别模型
- 在**二维空间**中，一条直线足以将两种类别完全分开
- 在**三维空间**中，需要用一张平面将两种类别分开
- **更高维空间**中，我们无法画出图形，可以将其称为“超平面”，用超平面来区分两种类别
- 假设训练数据集是线性可分的，感知机学习的目标是球的一个能够将训练数据集**正实例点和负实例点完全正确分开的分离线或分离面**
- 如果是非线性可分的数据，则最后无法获得**分离线或分离面**

3.2 支持向量机 (Support Vector Machine, SVM)

- SVM 是一种在特征空间上**使分类间隔最大**的分类器，它对两个类别进行分类
- 离分类直线**最近**的这些样本点对分类间隔大小非常“重要”，起着关键作用，这些样本点被称为**支持向量**。
- **几何间隔**：

$$d = \frac{|w_1x_1 + w_2x_2 + b|}{\sqrt{w_1^2 + w_2^2}} = y \times \frac{w_1x_1 + w_2x_2 + b}{\sqrt{w_1^2 + w_2^2}}$$

- **分类间隔**：分类间隔是支持向量几何间隔的两倍 $2d$
- **SVM 的目标**就是使得**分类间隔 $2d$ 最大**，分类间隔越大，分类效果越好

3.3 SVM 软间隔分类

- **软间隔 (Soft Margin) 分类**
  - 通常情况下，训练集中会存在一些**异常点**，而这些异常点会导致训练集线性不可分，但出去这些异常点之后，剩下的样本就是线性可分的。
  - 线性不可分意味着有些样本点的函数间隔是不能满足大于等于 1 的约束条件
- SVM 优化问题（软间隔）：对分错的点，给予惩罚

3.4 核函数

- 在有些情况下，图像线性不可分，但是我们仍然有办法能够进行线性分类。我们可以通过叫做"**核函数**"的映射，将低维的数据映射到高维空间中
- 原本在低维空间中线性不可分的样本，在**高维空间**中可能就线性可分了
- 通过这种方式，我们可以解决一些线性不可分的问题

3.5 多分类

- **方案 1： 1 vs all**
  - 训练类 A 与其他，类 B 与其他，类 C 与其他的二分类器
- **方案 2： 1 vs 1**
  - 训练类 A 与 B ， 类 B 与 C ， 类 C 与 A 的二分类器

3.6 性能分析

- **优点**:
  - 引入了最大间隔，分类精确度高
  - 样本量较小时，也能准确的分类
  - 核函数可以解决非线性分类
- **缺点**:
  - 样本量大时，计算速度慢
  - 多分类问题困难

4. 决策树分类

4.1 决策树介绍

- **决策树学习算法**一树形结构建立模型，类似于流程图，模型本身包含一系列逻辑决策，带有根据某一特性做出决定的决策节点。决策树从根节点开始，由叶节点结束，节点引出的分枝表示可做出的选择，叶节点表示遵循决策组合的效果
- 决策树是最广泛使用的机器学习技术之一，它菊虎可用于任何类型的数据建模，同时具有较好的性能

4.2 决策树构建

- **决策树构建**需要从根节点开始，判断实例的某个特征，根据这个特征的结果，将样本集划分为若干部分，然后以每一个部分再次作为根节点进行同样的操作
- 选取特征的时候，**需要这个特征能够将样本的类别尽量分开**，也就是说，这个特征对样本分类很明显
- 当一个节点里面的样本全部属于同一类别的时候，它就成了叶节点，此时每一个样本都对应着决策树里的一条路径

4.3 决策树特征选择

- **ID3 算法**
  - ID3 算法核心是在特征选择时计算每个特征产生的信息增益，选择**特征增益最大**的作为特征
- **C4.5 算法**
  - C4.5 算法在 ID3 上作了改进，使用**信息增益比**来判定特征
- **CART 算法**
  - CART 算法采用的是特征的取值来让树分枝，**一个节点只会分出两个树杈**，分别数属于特征 $A = a$ ，和不属于特征 $A = a$

4.4 数据类型

- 连续型数据：年龄……
- 离散型数据：学历、职业……
- 连续型数据离散化

4.5 性能分析

- **优点**:
  - 决策树易于理解和解释，可以可视化的分析，容易提取出规则
  - 可以同时处理标签型和数值型数据
  - 比较适合处理有缺失属性的样本
  - 数据测试、运行速度比较快
- **缺点**:
  - 容易发生过拟合
  - 不同的判定准则会带来不同的属性选择倾向

作业:

```
# 构建分类模型
print("支持向量机 SVM")
svm = SVC(C=1.0, kernel="linear")  # 创建SVM模型
svm.fit(X_train, y_train)  # 训练SVM模型
result(svm, X_test, y_test)  # 输出SVM模型的结果

print("K近邻 KNN")
knn = KNeighborsClassifier()  # 创建KNN模型
knn.fit(X_train, y_train)  # 训练KNN模型
result(svm, X_test, y_test)  # 输出KNN模型的结果

print("决策树 Decision Tree")
dt = DecisionTreeClassifier()  # 创建决策树模型
dt.fit(X_train, y_train)  # 训练决策树模型
result(svm, X_test, y_test)  # 输出决策树模型的结果
```