python 高级应用
决策树

2024 年 5 月 14 日

实验目标

以树为模型的分类器
编程语言：Python3
使用库：Numpy
分类任务：对个体的收入进行判断 >50K or $\leq 50K$
数据集：Adult (UCI Machine Learning Repository: Adult Data Set)

数据预处理

设计思路

使用 np.genfromtxt 函数读入训练数据，缺失项用众数填补。
分出 X,y
对 y 进行二值编码 y = y == y[0]
对 X 的非数值列进行数值编码。
测试数据同样按照上述方式操作，编码使用到的映射与训练数据一致。

读入数据

```
# 训练集
data = np.genfromtxt("adult/adult.data", delimiter=",", dtype=str)
data = data[~np.any(data == '?', axis=1)]
X = data[:, :14]
y = data[:, 14]
```

编码

```python
# 01编码
y = y == y[0]

# 数值编码
for i in [1, 3, 5, 6, 7, 8, 9, 13]:
    values = np.unique(X[:, i])
    encoding = {value: idx for idx, value in
        enumerate(values)}
    for j in range(X.shape[0]):
        X[j][i] = encoding[X[j][i]]

X = X.astype(int)
y = y.astype(int)
```

分割数据集

```
X_train = X[:X.shape[0] // 3 * 2, :]
y_train = y[:X.shape[0] // 3 * 2]

X_test = X[X.shape[0] // 3 * 2:, :]
y_test = y[X.shape[0] // 3 * 2:]
```

结点设置

```python
class TreeNode:
    def __init__(
            self, feature_index=None, threshold=
                None,
            value=None, left=None, right=None
    ):
        self.feature_index = feature_index
        self.threshold = threshold
        self.value = value
        self.left = left
        self.right = right
```

算法思路

采用递归建树的思路

```
function generate(X: array, y: array, depth: int)
    -> Node:
      找到X的最佳划分D
      按照D划分为X_l, X_r, y_l, y_r
      把划分数据交给左右子树,递归建树

      终止条件: y只剩下一类或达到最大递归深度
      返回值:  记录有划分标准D与左右孩子L,R的结点
```

参考代码

```python
def generateTree(self, X, y, depth=1):
        if depth == self.max_depth or len(set(y)) == 1:
            node = TreeNode(value=np.sort(y)[y.shape[0] >> 1])
        else:
            best_feature_index, best_threshold = find_best_split(X, y)
            left_indices = X[:, best_feature_index] <= best_threshold
            right_indices = ~left_indices

            left = self.generateTree(X[left_indices], y[left_indices], depth + 1)
            right = self.generateTree(X[right_indices], y[right_indices], depth + 1)

            node = TreeNode(
                feature_index=best_feature_index, threshold=best_threshold,
                left=left, right=right
            )
        return node
```

最佳划分

```python
def split_data(X, y):
    best_feature_index = None
    best_threshold = None
    best_entropy = inf

    for feature in X.features:
        D = set(X[feature])
        for v in D:
            left = i where X[i][feature] <= v
            right = i where X[i][feature] > v

            entropy = left  / y.shape[0] * entropy(y[left]) + right / y.shape[0] * entropy(y[right])

            if entropy < best_entropy:
                best_entropy = entropy
                best_feature = feature
                best_threshold = v

    return best_feature, best_threshold
```