南　开　大　学

计 算 机 学 院

机器学习实验报告

实验四 决策树分类器

姓名：王泳鑫

学号：1911479

年级：2019级

专业：计算机科学与技术

指导教师：卫金茂

2021 年 11 月 28 日

摘要

关键字：层次聚类，**Machine Learning**， **Deep Learning**

目录

一、 实验描述与要求

基本要求：绘制聚类前后样本分布情况

1. 基于 Watermelon-train1数据集（只有离散属性），构造ID3决策树；
2. 基于构造的 ID3 决策树，对数据集 Watermelon-test1进行预测，输出分类精度；

中级要求：

1. 对数据集Watermelon-train2，构造C4.5或者CART决策树，要求可以处理连续型属性；
2. 对测试集Watermelon-test2进行预测，输出分类精度；

二、 程序设计与代码实现

（一） ID3算法决策树构建

首先，ID3算法的核 思想应用信息增益准则作为标准,我们根据公式就可以实现计算信息熵，代码如下：

```python
# 计算信息熵
def calculate_entropy(y):
    log2 = math.log2
    unique_labels = np.unique(y)
    entropy = 0
    for label in unique_labels:
        count = len(y[y == label])
        p = count / len(y)
        entropy += -p * log2(p)
    return entropy
```

接下来，我们需要定义树的节点，代码如下：

```python
class DecisionNode():
    def __init__(self, feature_i=None, threshold=None,
                 value=None, true_branch=None, false_branch=None):
        self.feature_i = feature_i
        self.threshold = threshold
        self.value = value
        self.true_branch = true_branch
        self.false_branch = false_branch
```

然后要对特征进行划分，代码如下：

```python
    def divide_on_feature(X, feature_i, threshold):
    split_func = None
    if isinstance(threshold, int) or isinstance(threshold, float):
        split_func = lambda sample: sample[feature_i] >= threshold
    else:
        split_func = lambda sample: sample[feature_i] == threshold

    X_1 = np.array([sample for sample in X if split_func(sample)])
    X_2 = np.array([sample for sample in X if not split_func(sample)])

    return np.array([X_1, X_2])
```

接着，就是我们决策树的内容，代码如下：

```python
class DecisionTree(object):
def __init__(self, min_samples_split=2, min_impurity=1e-7,
             max_depth=float("inf"), loss=None):
    self.root = None  #根节点
    self.min_samples_split = min_samples_split
    self.min_impurity = min_impurity
    self.max_depth = max_depth
    # 计算值如果是分类问题就是信息增益，回归问题就基尼指数
    self._impurity_calculation = None
    self._leaf_value_calculation = None #计算叶子
    self.one_dim = None
    self.loss = loss

def fit(self, X, y, loss=None):
    self.one_dim = len(np.shape(y)) == 1
    self.root = self._build_tree(X, y)
    self.loss=None

def _build_tree(self, X, y, current_depth=0):
    """递归求解树

    """

    largest_impurity = 0
    best_criteria = None
    best_sets = None

    if len(np.shape(y)) == 1:
        y = np.expand_dims(y, axis=1)

    Xy = np.concatenate((X, y), axis=1)

    n_samples, n_features = np.shape(X)

    if n_samples >= self.min_samples_split and current_depth <= self.max_depth:
        # 计算每一个特征的增益值
        for feature_i in range(n_features):
            feature_values = np.expand_dims(X[:, feature_i], axis=1)
            unique_values = np.unique(feature_values)

            for threshold in unique_values:
                Xy1, Xy2 = divide_on_feature(Xy, feature_i, threshold)

                if len(Xy1) > 0 and len(Xy2) > 0:
                    y1 = Xy1[:, n_features:]
                    y2 = Xy2[:, n_features:]

                    # 计算增益值
                    impurity = self._impurity_calculation(y, y1, y2)

                    if impurity > largest_impurity:
                        largest_impurity = impurity
                        best_criteria = {"feature_i": feature_i, "threshold":
                                                  threshold}
```

```python
                            best_sets = {
                                "leftX": Xy1[:, :n_features],
                                "lefty": Xy1[:, n_features:],
                                "rightX": Xy2[:, :n_features],
                                "righty": Xy2[:, n_features:]
                            }

        if largest_impurity > self.min_impurity:
            true_branch = self._build_tree(best_sets["leftX"], best_sets["lefty"],
                                                current_depth + 1)
            false_branch = self._build_tree(best_sets["rightX"], best_sets["righty"],
                                                current_depth + 1)
            return DecisionNode(feature_i=best_criteria["feature_i"], threshold=
                                                best_criteria[
                "threshold"], true_branch=true_branch, false_branch=false_branch)

        # 计算节点的目标值
        leaf_value = self._leaf_value_calculation(y)

        return DecisionNode(value=leaf_value)


    def predict_value(self, x, tree=None):
        """预测

        """

        if tree is None:
            tree = self.root

        if tree.value is not None:
            return tree.value

        feature_value = x[tree.feature_i]

        branch = tree.false_branch
        if isinstance(feature_value, int) or isinstance(feature_value, float):
            if feature_value >= tree.threshold:
                branch = tree.true_branch
        elif feature_value == tree.threshold:
            branch = tree.true_branch

        return self.predict_value(x, branch)

    def predict(self, X):
        y_pred = []
        for x in X:
            y_pred.append(self.predict_value(x))
        return y_pred
```

最后是分类器的实现，代码如下：

```python
    class ClassificationTree(DecisionTree):
    def _calculate_information_gain(self, y, y1, y2):
        # 计算信息增益
        p = len(y1) / len(y)
```

```python
        entropy = calculate_entropy(y)
        info_gain = entropy - p * calculate_entropy(y1) - (1 - p) * calculate_entropy(y2)

        return info_gain

    def _majority_vote(self, y):
        most_common = None
        max_count = 0
        for label in np.unique(y):
            # 投票决定当前的节点为哪一个类
            count = len(y[y == label])
            if count > max_count:
                most_common = label
                max_count = count

        return most_common

    def fit(self, X, y):
        self._impurity_calculation = self._calculate_information_gain
        self._leaf_value_calculation = self._majority_vote
        super(ClassificationTree, self).fit(X, y)
```

（二） CART决策树构建

对于CART决策树，我们只需要把ID3算法中计算信息熵函数修改为计算基尼系数的函数，代码如下：

```python
def calculate_variance(X):
""" Return the variance of the features in dataset X """
mean = np.ones(np.shape(X)) * X.mean(0)
n_samples = np.shape(X)[0]
variance = (1 / n_samples) * np.diag((X - mean).T.dot(X - mean))

return variance
```

同时把分类器中计算函数修改为：

```python
def _calculate_variance_reduction(self, y, y1, y2):
var_tot = calculate_variance(y)
var_1 = calculate_variance(y1)
var_2 = calculate_variance(y2)
frac_1 = len(y1) / len(y)
frac_2 = len(y2) / len(y)

# 使用方差缩减
variance_reduction = var_tot - (frac_1 * var_1 + frac_2 * var_2)

return sum(variance_reduction)
```

三、 测试与结果演示

（一） ID3决策树

通过测试数据集来测试，代码如下：

```
from sklearn.metrics import accuracy_score

x_train = np.loadtxt('D:\\Data\\lab6\\Watermelon-train1.csv',dtype='str',delimiter=','
                                            ,skiprows=1,usecols=(1,2,3,4))
y_train = np.loadtxt('D:\\Data\\lab6\\Watermelon-train1.csv',dtype='str',delimiter=','
                                            ,skiprows=1,usecols=(5))

x_test = np.loadtxt('D:\\Data\\lab6\\Watermelon-test1.csv',dtype='str',delimiter=',',
                                            skiprows=1,usecols=(1,2,3,4))
y_test = np.loadtxt('D:\\Data\\lab6\\Watermelon-test1.csv',dtype='str',delimiter=',',
                                            skiprows=1,usecols=(5))

clf = ClassificationTree()
clf.fit(x_train,y_train)


y_pred = clf.predict(x_test)

accuracy = accuracy_score(y_pred,y_test)

print(accuracy)
```

结果如图2所示

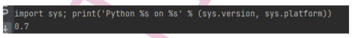

图 1: ID3决策树结果

（二） CART决策树

这里需要注意的是我们需要把密度从字符串形式转换为float类型，代码如下：

```
from sklearn.metrics import accuracy_score

x_train = np.loadtxt('D:\\Data\\lab6\\Watermelon-train2.csv',dtype='str',delimiter=','
                                            ,skiprows=1,usecols=(1,2,3,4,5))
y_train = np.loadtxt('D:\\Data\\lab6\\Watermelon-train2.csv',dtype='str',delimiter=','
                                            ,skiprows=1,usecols=(6))

for i in range(17):
    x_train[i][4]=float(x_train[i][4])

x_test = np.loadtxt('D:\\Data\\lab6\\Watermelon-test2.csv',dtype='str',delimiter=',',
                                            skiprows=1,usecols=(1,2,3,4,5))
y_test = np.loadtxt('D:\\Data\\lab6\\Watermelon-test2.csv',dtype='str',delimiter=',',
                                            skiprows=1,usecols=(6))

for i in range(5):
    x_test[i][4]=float(x_train[i][4])

```

```python
clf = RegressionTree()
clf.fit(x_train,y_train)


y_pred = clf.predict(x_test)

accuracy = accuracy_score(y_pred,y_test)

print(accuracy)
```

结果如图2所示

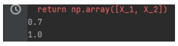

图 2: CART决策树结果