hw3 decision tree

21302010042 侯斌洋

1. 数据分析

- span_pub.csv共有 4142 行，59 列，其中第一列为序号列，中间 57 列为特征列，最后一列为标签列。
- 特征列均为连续数值属性，标签列均为bool属性。
- 共有 4141 记录，其中 2521 个正样本，1620 个负样本。
- 数据中不存在缺失值，正负样本分布也较为均匀，适合用于决策树的训练。

2. 代码说明

2.1 CART树实现

```
class Node:...

def _split(feature, threshold, X, y):...

def _most_common_label(y):...

class Dtree:...
```

2.2 main.py

- 新增两个参数`min_samples_split`和`max_depth`，分别表示最小分裂样本数和最大深度。参数的默认值为grid_search得到的最佳值。

```
def main(X: list, Y: list, test_x: list, min_samples_split=5, max_depth=10) ->
list:
    cart = Dtree(min_samples_split, max_depth)
    cart.fit(np.array(X), np.array(Y))
    return cart.predict(np.array(test_x))
```

2.3 其他代码

```
# 读取数据
def load_csv(filename):...

# 划分训练测试集，默认训练集占70%，测试集占30%，每次划分时都随机打乱数据
def spit_data(data, test_rate):...

# 网格搜索,用于寻找最佳参数
def grid_search(data, test_rate):...

if __name__ == '__main__':
```

```
    # 读取数据
    data = load_csv('./data/span_pub.csv')
    # 测试集占30%
    test_rate = 0.3

    # grid_search(data, test_rate)

    # 划分训练测试集
    train_X, train_Y, test_X, test_Y = spit_data(data, test_rate)
    print(f'train_data_len: {len(train_X)}')
    print(f'test_data_len: {len(test_X)}')

    # 训练模型并预测
    predict_Y = main(train_X, train_Y, test_X)

    # 计算准确率
    count = 0
    for i in range(len(predict_Y)):
        if predict_Y[i] == test_Y[i]:
            count += 1
    print(f'accuracy: {count / len(predict_Y)}')
```

3. 运行结果

- 网格搜索的结果保存在`search_result.log`中，最佳参数为`min_samples_split=5`和`max_depth=10`。
  以下为使用最佳参数训练的结果：

```
-----------------------------------------
min_samples_split: 5, max_depth: 10
accuracy0: 0.919549477071601  time: 248.8458924293518
accuracy1: 0.915526950925181  time: 285.98415994644165
accuracy2: 0.910699919549477  time: 292.7223846912384
average_accuracy: 0.9152587825154197
-----------------------------------------
```

4. 优化

- CART树的实现按照标准的CART算法。
- 优化上主要采用预剪枝的方式，通过调整`min_samples_split`和`max_depth`参数来控制树的复杂度，避免过拟合。
- 网格搜索是机器学习中常用的调参方法之一。代码中也提供了网格搜索的实现，可以在不同的数据集上寻找最佳参数。
- 代码中`main.py`的默认参数为网格搜索得到的最佳参数。最终得到的准确率为`91%`左右。