决策树机器学习实验——171250574杨逸存

实验简介

- 本实验使用python语言及numpy、pandas和matplotlib等python库实现了**ID3决策树**算法
- 实验数据为给定隐形眼镜的小量数据集，构造决策树预测患者佩戴隐形眼镜的类型，并通过Matplotlib绘制树图形。

算法/代码详解

ID3决策树算法

- ID3算法通过特征的信息增益**Gain(D, A) = H(D) - H(D|A)**

  其中**H(D|A) = ∑ pi*H(D|A=ai)，H(D) = -∑pi*log(pi)** 来实现最有特征的选择

- 算法输入：训练数据集D，特征集A，阈值ε；输出：决策树T

  1. 若当前训练集D中所属实例属于同一类或当前特征集A为空，则返回单节点，节点值为实例所属类或实例数最多的一个类。
  2. 否则，计算A中个特征对D的信息增益，选择信息增益最大的特征Ak。
  3. 如果Ak的信息增益也小于阈值ε，则同样返回值为实例数最多的类的一个单节点。
  4. 否则，对Ak的每一种可能值ai，依Ak=ai将D分割为若干非空子集Di。
  5. 对每个Di，以A - {Ak}作为特征集合，递归调用Step1~4，返回的子树作为原树的一个分支，并返回原树T；

核心代码

- 决策树生成主方法解析

```
decision_tree = generate_decision_tree(data_set, eps=0.0001)
```

使用dict字典数据结构模拟树，其中子树为嵌套字典对象，字典键为节点值或连接分支值，叶子节点为最终分类对象（**具体见实验结果分析**）

前面的if分支主要解决递归结束特殊条件，后半部分的for循环中递归调用generate_decision_tree，返回的子树作为**当前节点特征的一个具体取值分支**下的子树。

```
def generate_decision_tree(data_set: pd.DataFrame, eps: float):
    '''
    生成决策树，迭代调用
    :param data_set: 当前数据集
    :param eps: ε信息增益阈值
    :return: 字典（子树）或单个值（叶子）
    '''
    # 分类结果列表
```

```
    class_list = np.array(data_set['class'])
    # 如果所有实例属于同一个类
    if sum(class_list == class_list[0]) == len(class_list):
        return class_list[0]
    # 如果标签集为空（只剩class列），则为单节点树，把实例数最大的类作为此节点标记类
    if len(data_set.columns.values) == 1:
        return get_most_common_class(data_set)
    # 获取信息增益最大的特征及其增益
    highest_gain_feature, highest_gain = get_feature_with_highest_Gain(data_set)
    # 增益小于ϵ，单一节点，返回实例数最大的类
    if highest_gain < eps:
        return get_most_common_class(data_set)
    # 构建树
    decision_tree_dict = {highest_gain_feature: {}}
    # 对每个最高增益特征的取值进行分割数据集，并进行递归调用生成树
    feature_values = set(data_set[highest_gain_feature])
    for one_value in feature_values:
        # 分割Di
        divided_data_set = data_set[data_set[highest_gain_feature] == one_value]
        # 去除列，A = A - {Ak}
        divided_data_set = divided_data_set.drop(labels=highest_gain_feature, axis=1)
        # 生成子树
        decision_tree_dict[highest_gain_feature][one_value] = generate_decision_tree(divided_data_set, eps)
    return decision_tree_dict
```

- 信息熵、条件熵计算

  信息熵H(D) = -∑pi*log(pi)

```
def H(data_set: pd.DataFrame):
    '''
    计算经验熵H(D)
    :param data_set: 数据集D
    :return:
    '''
    class_list = data_set['class']
    class_values = set(class_list)
    H = 0.0
    for one_value in class_values:
        this_value_list = data_set[data_set['class'] == one_value]
        p_i = float(len(this_value_list)) / float(len(data_set))
        H -= p_i*np.log2(p_i)  # H(X) = -Σpi*log(pi)，取2为底的对数
    return H
```

在**条件熵**计算中，传入特征取某一具体值的子集合，**调用信息熵计算方法**，简化了逻辑计算代码

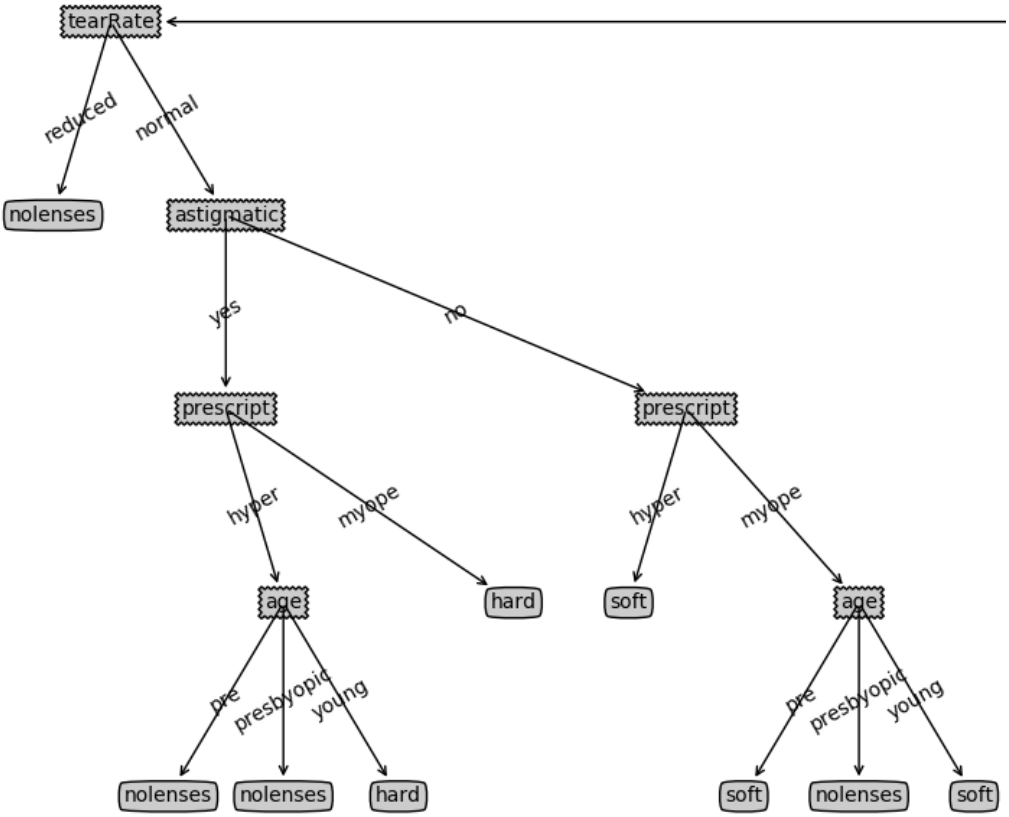
```python
def H_condition(data_set: pd.DataFrame, feature: str):
    '''
    计算经验条件熵H(D|A)，H(D|A) = Σ pi*H(D|A=ai)
    :param data_set: 数据集D
    :param feature: 特征A
    :return:
    '''
    class_list = data_set['class']
    class_values = set(class_list)
    feature_values = set(data_set[feature])
    H_con = 0.0
    for one_feature_value in feature_values:
        this_value_list = data_set[data_set[feature] ==
one_feature_value]
        pi = float(len(this_value_list)) / float(len(data_set))
        H_con_this_value = H(this_value_list)
        H_con += pi * H_con_this_value
    return H_con
```

实验结果分析

- 实验数据为助教给的lenses.txt文件，有24个样本数据量
- 生成的决策树：

```
{'tearRate':
    {'normal':
        {'astigmatic':
            {'no':
                {'prescript':
                    {'hyper': 'soft',
                     'myope':
                        {'age':
                            {'pre': 'soft',
                             'presbyopic': 'nolenses',
                             'young': 'soft'}}}},
             'yes':
                {'prescript':
                    {'hyper':
                        {'age':
                            {'pre': 'nolenses',
                             'presbyopic': 'nolenses',
                             'young': 'hard'}},
                     'myope': 'hard'}}}},
    'reduced': 'nolenses'}
}
```

- 调用TreePlotter.py，修改一下figure绘图参数，得到以下树图形：


tearRate
reduced
normal
nolenses
astigmatic
yes
no
prescript
prescript
hyper
myope
hyper
myope
age
hard
soft
age
pre
presbyopic
young
pre
presbyopic
young
nolenses
nolenses
hard
soft
nolenses
soft