ID3决策点总结

一. ID3算法如何选择最优的特征进行划分：

1.计算整体样本的信息熵

2.通过**特征索引，特征取值**对样本集进行划分

3.计算信息增益并获得增益最大的**特征索引**

二.tree.py 中的三个函数

1.calc_shannon_ent :传入data,计算ent

2.split_data :传入data,index,value,计算sub_data

3.get_best_split :传入data,计算best_index

4 递归建树

所谓递归，简单来说，就是一个函数直接或间接调用自身的一种方法。它通常把一个大型复杂的问题层层转换为一个与原问题相似的规模较小的问题

问题可用递归解决具备的两个条件：

1. 子问题需与原问题为同样的事情，且规模更小
2. 程序具备终止条件

决策树的建立

决策树问题可用递归解决具备的两个条件：

1. 每次选择最优特征划分后得到的多个sub_data子样本集比原data规模较小的样本集，并且也是要找到最优特征进行分割
2. 样本是有限数量的，每次划分的数量都会减少，肯定是能够终止的

4. 1终止条件

终止条件:(从决策点变成叶子节点)

1. 如果所有样本的标签全部相同或者只剩下一个样本：

   将该标签作为叶子节点。

2. 如果所有特征已经用完(有多个样本，只剩标签列，但标签仍然不统一：

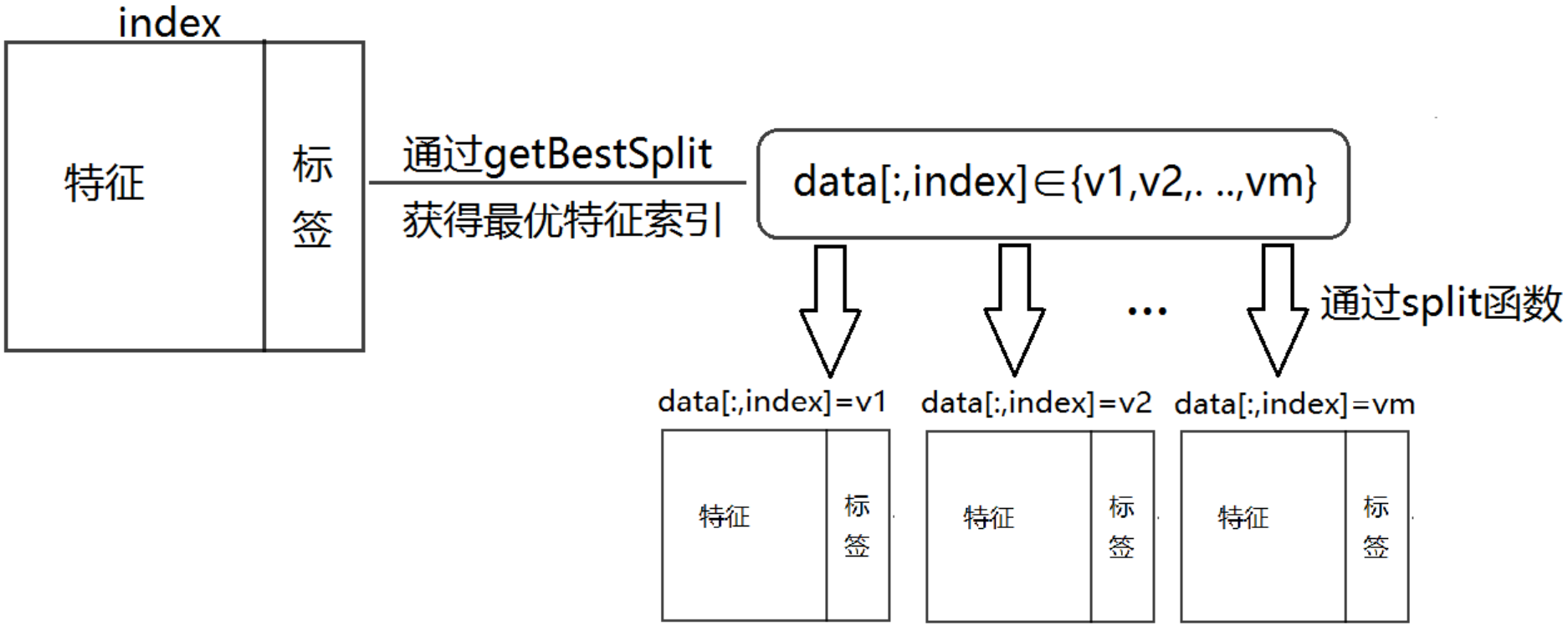
   将出现次数最多的标签作为叶子节点

需要一个**函数to_leaf_node()**：输入一个标签行label_list, 输出出现次数最多的label

递归建树的过程中需要维护一个特征**索引列表f_index**, 每次都将最优特征索引删除

4.2 递归过程

递归过程:如何将大规模问题分割成小规模问题
回顾一下前三个函数:(打开tree.py文件)



4.3 决策点结构

决策点结构:什么形式保存一棵树，保存哪些信息

1. 使用字典来保存 tree={}

2. 保存决策点的特征索引，**以及该特征空间对应的多棵子树**(子树有多少棵，由特征空间长度决定)

   1. tree={‘index’:index,’child’:{}}

   2. 每一棵子树怎么确定，假设特征取值v1,
      所得子样本集为sub_data的子树
      tree[‘child’][v1]=create_tree(sub_data)

create_tree函数为建树函数

4.4 建树步骤

index_list=[i for i in range(len(data[0]))] #索引列表

create_tree(data,index_list):

判断是否符合终止条件，如果是：

返回类别标签作为叶节点

通过get_best_split获得b_index#(b_index是相对值)

获得该特征索引对应的特征空间

利用最优索引获得索引index#(最优索引是个相对值)

将索引从index_list中删除

创建树tree={'index':index , 'child':{}}

对应特征空间的每个值value：

利用split获得子样本集sub_data

tree['child'][value]

=creat_tree(sub_data,index_list) #(子树递归)

4.5 建树步骤

```
create_tree(data,index_list):
    label_list=[d[-1] for d in data]
    if (len(label_list)==1 or len(set(label_list))==1)：#红色的可以删
        return label_list[0]
    if(len(data[0])==1):
        return to_leaf_node(label_list)
```

```
    b_index=get_best_split(data)
    fea_space=set([d[b_index] for d in data])
```

```
    index=index_list[b_index]
    del(index_list[b_index])
```

```
    tree={'index':index,'child':{}}
    for value in fea_space:
        sub_data=split(data , d_index , value) #注意是d_index
        tree['child'][value]=create_tree ( sub_data , index_list )
```

4.6 ID3建树练习

ID	年龄	有工作	有自己的房子	信贷情况	类别
1	青年	否	否	一般	否
2	青年	否	否	好	否
3	青年	是	否	好	是
4	青年	是	是	一般	是
5	青年	否	否	一般	否
6	中年	否	否	一般	否
7	中年	否	否	好	否
8	中年	是	是	好	是
9	中年	否	是	非常好	是
10	中年	否	是	非常好	是
11	老年	否	是	非常好	是
12	老年	否	是	好	是
13	老年	是	否	好	是
14	老年	是	否	非常好	是
15	老年	否	否	一般	否

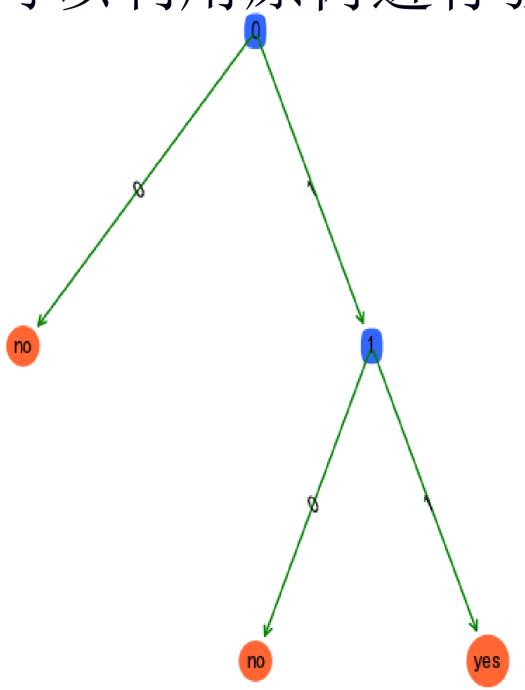
5 使用ID3算法进行分类预测

{'index': 2, 'child': {'是': '同意', '否': {'index': 1, 'child': {‘是’: ‘同意’, ‘否’: ‘拒绝’}}}} #通过treePlot2.py画图

通过分支找到对应的叶子节点：

{'index': 0, 'child': {0: 'no', 1: {'index': 1, 'child': {0: 'no', 1: 'yes'}}}}

下面的例子其实为了能将字典放在一行里，使用了另外棵树：可以利用原树进行验证



叶子节点和决策节点的区别：

叶子节点对应的子树是一个**值**

决策点对应的子树是一个**字典**

待测试的样本：

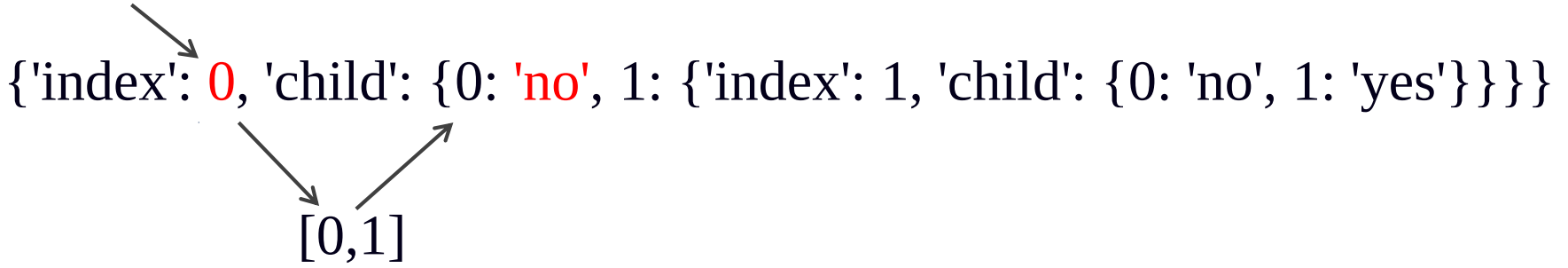
[0, 1]

[1, 0]

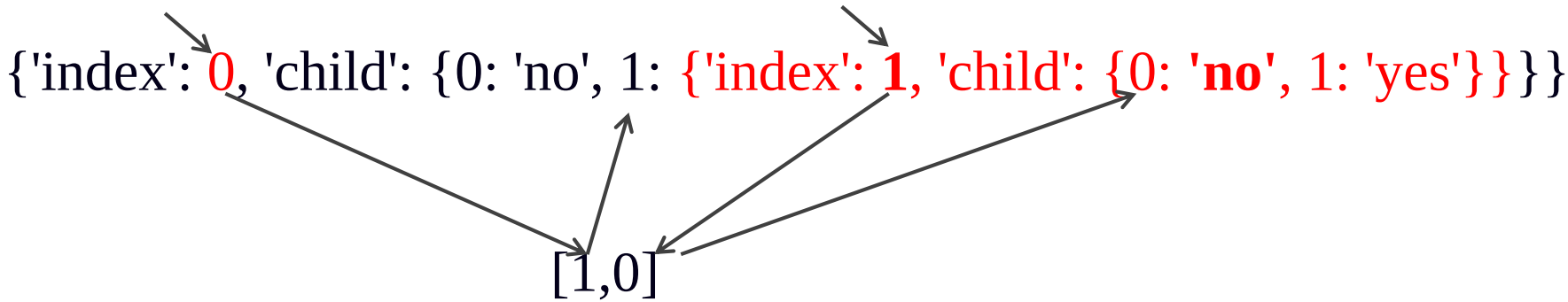
[1, 2]

如何确认他们的分类？

5.1 递归预测



1.通过tree找到index
2.通过example找到第index个特征的值value
3.通过tree中取值为value的子树获得类别标签



3.如果第三步找到的value子树，依旧是一个字典(isinstance)：
4.将tree的value子树作为tree，重复以上步骤

5.2 提高算法的稳定性：

预测样本点 [1,2] 的分类情况程序就会报错。

因此在第2步中会检测该值对应的子树是否存在
也就是：
判断value是否存在于tree[‘child’]字典中的键值中

5.3 算法代码

```
predict(tree,example):
    index=tree[‘index’]
    value=example[index]
    if(value not in tree[‘child’].keys()):
            return ‘error’
    if(isinstance(tree[‘child’][value],dict)):
            tree=tree[‘child’][value]; predict(tree,example)
   return tree[‘child’][value]
```

总结

一. 如何通过标签判断一个样本集的不纯度：信息熵

二. ID3算法如何选择最优的特征进行划分：
条件熵和信息增益

1. 计算整体样本的信息熵
2. 通过特征索引，特征取值对样本集进行划分
3. 计算条件熵并进行累加
4. 计算信息增益并获得最大增益与对应的特征索引

总结

三. ID3算法递归建树：终止条件、递归过程、树结构

1. 要维护一个特征索引列表
2. 要有一个决策节点变叶子节点的函数
3. 保存的树信息包括：

{特征索引：___，

子节点：{以值为键的多棵子树}}

四. ID3进行预测：通过递归找到叶子节点

ID3 算法的不足:

1. 只能处理离散型性特征
2. 只能处理分类问题