**使用决策树预测英雄联盟胜负**

**PB20000156 徐亦昶**

Part 1 主要算法

决策树的训练算法流程如下：

对于要训练的节点node，设训练数据集为D，属性集为A:

如果它是叶子节点，直接返回;

if D中样本全属于同一类别C then

将node标记为C类叶节点;return

end if

if A为空 or D中样本在A上取值相同 then

将node标记为叶节点，其类别标记为D中样本数最多的类;return

end if

从A中选出最优划分属性a;

if a为离散型 then

for a的每一个值a\* do

为node生成一个分支（即代码中的child）;令D\*表示D中在a上取值 为a\*的样本子集;

if D\*为空 then

将分支节点标记为叶节点，其类别标记为D中样本最多的类;return

else

令D\*为该分支的训练数据集;

end if

end for

else

为node生成两个分支;

依次考虑对应属性值大于和小于划分属性a的阈值的样本子集:

如果子集为空，则将分支节点标记为叶节点，其类别标记为D中样 本最多的类;return

否则令对应样本子集为该分支的训练数据集;

end if

训练上面生成的每一个数据集

最优划分属性的选择：

如果节点的深度超过最大深度，直接返回None，不再进行选择;

for clsprop in self.clsproperties:(依次遍历所有可选择的划分属性)

if clsprop为离散型:

计算信息增益和信息增益率

end if

if clsprop为连续型:

将训练样本去重并排序;

if 去重后仅剩一个元素 then

continue

end if

依次取两个相邻元素，取它们的中点作为阈值，取使得信息增益最大的 那个作为最后分类器的阈值;

end if

在信息增益高于平均的分类器里选出信息增益率最大的那个作为该节点的分类器

剪枝算法：

采用后剪枝

prune(node):

if node为叶节点 then

return

end if

for child in node.child:

prune(child);

将node设成叶节点，其类别标记为训练样本中样本最多的类;

计算此时决策树的精确度;

如果精确度比原来高，则剪枝，否则将node改回中间节点;

Part2 使用方法

交叉验证参见代码predict.py

对于非交叉验证，第一步是使用file\_utils中的函数read\_data，输入为要截取的数据在数据集中的范围，即第几行到第几行，返回types(每个属性的类型，即连续还是离散)，properties(属性)，如：types,properties=read\_data(0,10000)将读取前1000行的数据。

接着生成一个数据集，即Data类：test\_data=Data(types,properties)，再用同样的方法生成测试数据集。

然后生成一个DTree类：tree=DTree(train\_data,test\_data,np.array([0,1]))，其中最后一个参数是要把所有属性所属类的集合，不要重复。

接下来训练决策树：tree.train()，调节prune=False可使决策树不剪枝。

最后即可对结果进行预测。除了tree.compute\_accuracy()外，还可以用tree.predict()进行预测。该函数有两个重载，对应两种参数，一种是输入一个列表，依次列出各属性的值（表示数据所属类的属性class跳过）；另一种是一个字典，键是属性名，值是属性对应的值，该字典允许包含样本中不存在的属性。通常使用决策树时用第一个重载，因为不需要输属性名，但是开发者测试时使用第二个重载，这样可以直接利用以生成的字典形式的属性。

Part3 模型性能

模型一开始是未剪枝的，按训练：测试=4：1对全部80000个数据进行测试，测得精确度为94.725%。

然而，不剪枝可能导致过拟合，所以后来给决策树提供了剪枝模块（可选），以提高模型的泛化性能。

剪枝后按照训练数据：测试数据=4：1对全部80000个数据进行交叉验证，得5次测试的精确度分别为95.7875%，95.7875%，95.75625%，95.5875%，95.96875%，平均精确度为95.7775%，可见剪枝提高了模型的泛化性能。

Part4 编写过程遇到的问题

1. np.append只是返回一个副本，并不会在原来的array上追加元素，开始不知道这一点，导致有个数组被append后还是空的
2. 忽略“self."
3. numpy中空数组为None，不能用len==0检测
4. 应先判断数组是否为空再操作
5. 在处理连续型数据仅有一个值时，方法过于复杂，其实

if len(sorted\_data)==1: #该属性的值相同

continue

就行了。

1. 测试数据时遇到KeyError 原因：对DataFrame按列索引得到的是Series，它的index是从原数据的行号开始的。需要使用.values将Series转换成ndarray。
2. 返回的分类器不是最好的 原因：误认为compute\_classifier方法中的gain是按大小排序的。
3. 有时遇到child[1]数组越界 原因：对连续型变量分类时，只计算了阈值不小于classifier.thresh的部分。