文本分类作业报告

邵智轩　　1400012141

2018 年 3 月 15 日

1　程序目标

用 ID3 算法构建一棵决策树，实现文本分类。数据来源为中文微博数据集（已分词）。用 Python 实现，不能直接调决策树算法库。

文本来源于 9 个文件夹，每个文件夹自成一类，分别为：[" 财经"," 房产"," 健康"," 教育"," 军事"," 科技"," 体育"," 娱乐"," 证券"]

2　实现方法

2.1　数据预处理

2.1.1　集合化：

将读入的文本（str）转为集合（set），集合元素为文本中出现过的词。这一过程丢失了一些信息：忽略了同一文本中词的出现次数；忽略了词的前后顺序。我认为这些信息相对次要，而这一处理可大大简化后续算法，使得每一词成为一项属性，而属性的取值只有 2 种（有/无）。

2.1.2　数字匿名化

把所有数值（如'2'，'2.5' 等）都视为同一个词，记为'1'。这一步需要先判断某一字符串是否属于数值，由程序中的 is_num(str) 函数实现。

2.1.3　选取属性

我设置了一个总出现次数的下限 least_frequency，将总出现次数大于等于它的词都作为属性。以它为结点对 set 分类时，只分有或无两类。

2.2 构建决策树

2.2.1 交叉熵

如果训练样本 S 在第 i 类中的比例为 p_i，其交叉熵

$$Entropy(S) = -\sum_{i=1}^{n} p_i \log p_i$$

在程序中由 `cross_entropy(distr)` 函数计算。

2.2.2 信息收益

S 是分类前的训练样本集合，A 是一项属性，即一个词。以文本集合中有无该词为标准来划分样本，信息收益为：

$$Gain(S, A) = Entropy(S) - \frac{|S_{A=1}|}{|S|} Entropy(S_{A=1}) - \frac{|S_{A=0}|}{|S|} Entropy(S_{A=0})$$

在程序中由 `information_gained(init_distr, selected_distr)` 计算。

2.2.3 递归生成决策树

由以下函数实现：

```
def DecisionTree_Building(data,selected_vocab,least_info_gained,
  stopping_proportion) -> int, list, None
  ''' 递归构建决策树
  data: 当前训练样本集
  selected_vocab: 可选的属性
  least_info_gained: 最小信息收益
  stopping_proportion: 最小停止比例
  '''
  number_by_class=np.array([len(cls) for cls in data])# 每个分类下的样本数
  if number_by_class.max()/number_by_class.sum()>=stopping_proportion:# 判停条件 1: 超过
      return int(number_by_class.argmax())
  count_by_class=[count_class(data[cls]) for cls in range(len(data))]
  vocab_distr=integrated_count(selected_vocab,count_by_class)
  (selected,most_info_gained)=most_gained_property(
```

```
                                number_by_class,vocab_distr)
    if most_info_gained<least_info_gained:# 判停条件 2: 继续分类已经难以增加新的信息了
        return int(number_by_class.argmax())
    (data_with,data_without)=devide_by_property(data,selected)# 将数据分为两堆
    del selected_vocab[selected]# 不用再考虑这个属性了
    tree=[]
    tree.append(selected)
    tree.append(DecisionTree_Building(data_with,
                selected_vocab,least_info_gained,stopping_proportion))# 添加左子树
    tree.append(DecisionTree_Building(data_without,
                selected_vocab,least_info_gained,stopping_proportion))# 添加右子树
    if type(tree[1])==str and tree[1]==tree[2]:return tree[1]# 后面的分叉导出相同的结果
    return tree
```

若不满足判停条件，在剩余属性中选择信息收益最大的属性（词 `A`），将训练样本集一分为二，即有这个词的和没有这个词的。将这个词从可选属性中删除，以这个词为结点，分别**以有这个词的样本集**和**无这个词的样本集递归**添加左子树 `Left` 和右子树 `Right`。返回列表 `[A,Left,Right]`。

判停条件：

1. 若当前训练样本集中超过某一比例的样本都在某一类，则直接返回该类作为叶结点。这一比例为可调参数 `stopping_proportion`。
2. 若所有可选属性的信息收益都小于某一值（没有显著的统计意义），则直接返回当前训练样本集中比例最高的类。这一值设为 `least_info_gained`。

2.2.4 决策树的数据结构

由上一节的算法看出，决策树采用列表嵌套结构，即 `[A,Left,Right]`。`A` 是划分用的属性（词），若 `Left` 或 `Right` 是一个 `int`，则为叶结点；若 `Left` 或 `Right` 是一个 `list`，则仍为子树。

3 结果分析

本实验中可调参数有 3 个,分别为 `least_frequency`,`least_info_gained` 和 `stopping_proportion`。通过 10-fold Cross-Validation，确定出大致最优的参数设置为：(在这些参数附近变动，结果差异不大)

```
least_frequency=5
least_info_gained=0.005
stopping_proportion=0.9
```

更详细的结果分析见下。

1. 备选属性个数：经过前述预处理后，总计不同的词个数为 57692。设置 `least_frequency=5` 后，即只考虑总出现次数不少于 5 次的词，备选属性个数为 12293。
2. 树的规模：结点个数为 2750，树的高度为 860。
3. 建树效率由于属性个数很多，树的高度和结点数很多，建一棵树通常需要较长的时间。当参数设置如前时，用全样本建树在我的电脑上需要 40 min — 1 h。
4. 正确率

a) 用全样本集合训练，并在从全样本集中抽出 1/10 做测试集，正确率为：0.835
b) 10-fold Cross-Validation，平均正确率为：0.681

5. 混淆矩阵下面显示了 10-fold CV 的平均混淆矩阵。行表示样本实际来源于哪一类，列表示样本被分到哪一类。显然，对角线元素表示每一类样本的分类正确率。

	财经	房产	健康	教育	军事	科技	体育	娱乐	证券
财经	0.661	0.1263	0.0193	0.0088	0.0105	0.0826	0.017	0.0269	0.0467
房产	0.470	0.3955	0.0132	0.0099	0.0083	0.0264	0.030	0.0264	0.0206
健康	0.188	0.0284	0.6522	0.0090	0.0060	0.0179	0.052	0.0448	0.0015
教育	0.149	0.0383	0.0273	0.5758	0.0042	0.0182	0.062	0.1221	0.0023
军事	0.118	0.0127	0.0025	0.0013	0.8091	0.0063	0.021	0.0291	0.0000

科技 0.296 0.0293 0.0093 0.0079 0.0172 0.5473 0.026 0.0308 0.0365
体育 0.062 0.0096 0.0066 0.0051 0.0045 0.0063 0.838 0.0665 0.0006
娱乐 0.121 0.0169 0.0160 0.0147 0.0058 0.0186 0.086 0.7167 0.0044
证券 0.241 0.0301 0.0069 0.0025 0.0077 0.0702 0.010 0.0086 0.6228

可以得到一些有趣的结论。如分类正确率超过 80% 的类有体育（83.8%）和军事（80.9%），这说明这两类文本的特征明显；分类正确率最低的为房产（39.6%），这说明这类文本易于其他类混淆，有时难以根据词来区分开。可以看到，房产、科技类下有很多样本被分到财经（47.0% 和 29.6%）类。

在当前算法下，一方面由于财经文本的词最多，另一方面排在列表第一个，使得决策树算法在执行时不能确定的文本最终都归到了财经类下。

6. 参数选择在选择最优参数前，做了一些实验，改变各个参数看 10-fold CV 的平均正确率，结果如下：

least_frequency	least_info_gained	stopping_proportion	Accurate Rate
3	0.005	0.9	67.9
4	0.005	0.9	67.7
5	0.003	0.9	68.0
5	0.005	0.9	68.1
5	0.008	0.9	67.8
6	0.005	0.9	67.6
10	0.005	0.85	67.6
10	0.005	0.9	67.8
20	0.01	0.9	34.1

可以看到，在最优参数值附近做变动，正确率变化较小。