实验五 决策树

一、决策树是什么

首先，在熵的概念之下，分类在做些什么事情？熵，指的是在自然条件之下，事物的混乱程度，熵越大，事物的混乱程度越大，我们认为这越接近于现实。而当我们将每一件物品都分好类，同类的放在一起，那么这个时候事物就不再混乱了，熵就会减小，也就是相对的越接近于人工划分。（记住这个理解，有助于理解以下的决策方法）

决策树(Decision Tree）是在已知各种情况发生概率的基础上，通过构成决策树来求取净现值的期望值大于等于零的概率，评价项目风险，判断其可行性的决策分析方法，是直观运用概率分析的一种图解法。由于这种决策分支画成图形很像一棵树的枝干，故称决策树。在机器学习中，决策树是一个预测模型，他代表的是对象属性与对象值之间的一种映射关系。熵=系统的凌乱程度，使用算法 ID3, C4.5 和 C5.0 生成树算法使用熵，基于信息学理论中熵的概念，而算法 CART 使用 GINI 系数。决策树是一种有监督学习模型。

i. 它的组成可分为：

——决策点，在决策点中，我们计算最优决策属性。

——枝，根据决策点属性的不同值的划分之后，引出决策枝条，每个枝条代表着决策点属性的一种值。

——结果节点，当数据只剩下一类的时候，或者停止分类的时候，我们根据某种自定义的方法决定到达这个节点的数据最终归为哪个类。在本实验，推荐用多数投票算法（Majority Vote Algorithm）的方式决定。

ii. 决策树的特点

决策树直接体现数据的特点，直接根据数据划分得到一个模型，易于理解且实现不难，因此只要有理想的大量的数据，决策树的效果是非常优秀的。决策树利于知识库（Knowledge base）的构建，例如一些简单的“智能”机器人。

但是，决策树对于连续性的数据比较难以构建，需要找到最优的连续数据表达为离散数据的方法（具体的方法再后文会加以讨论），而当类别太多的时候，决策的效果也会下降，当数据的属性太少，也不能很理想地构建的决策树。

综上所述，决策树不是一个当下热门的模型，要求大家掌握原理，学会基本的实现即可。

根据某种策略对训练数据进行划分之后形成的决策模型

二、决策树的原理

ID3

决策策略：信息增益（Information Gain），假设有数据类标签 D 和数据其一属性 A 计算式如下：

$$g(D,A) = H(D) - H(D|A)$$

其中，

$$H(D) = -\sum_{d\in D} p(d)\log p(d),\quad \text{D 的熵}$$

$$H(D|A) = \sum_{a\in A} p(a)H(D|A=a)\ ，\text{在 A 条件下 D 的熵，即条件熵}$$

怎么理解信息增益呢？在开篇提过，熵，指的是数据的混乱程度，数据越混乱越接近与自然条件下的分布。那么 H(D)指的是当前数据的熵，而 H(D|A)假定了在 A 条件下 D 的熵，也就是假设我们用 A 将 D 归一下类，得到的数据子集的熵的总和。那么当条件熵 H(D|A)越小，我们划分出来的数据的类别越清晰，越合理。那么这个属性在当前数据中是最优决策属性。原来的熵减去分类之后的条件熵，我们称为信息增益，也就是我们这样分类之后得到的信息量有多大。

信息增益越大，优先选择作为决策根节点

C4.5

决策策略：信息增益率（Information Gain ratio），假设有数据类标签 D 和数据其一属性 A 计算式如下：

$$gRatio(D,A) = (H(D) - H(D|A))/SplitInfo(D,A)$$

其中，

$$SplitInfo(D,A) = -\sum_{j=1}^{v}\frac{|D_j|}{|D|}\times\log(\frac{|D_j|}{|D|})$$

信息增益率越大，优先选择作为决策根节点

为避免某个属性的不同值的个数过多，造成信息增益增大。

CART

决策策略：GINI 系数（Gini Index），假设有数据类标签 D 和数据其一属性 A 计算式如下：

$$gini(D,A) = \sum_{j=1}^{v} p(A_j)\times gini(D_j|A=A_j)$$

v 指属性 A 的种类个数，j 指属性 A 的第 j 个种类。

其中，

$$gini(D_j|A = A_j) = \sum_{i=1}^{n} p_i(1 - p_i) = 1 - \sum_{i=1}^{n} p_i^2$$

n 指有多少个标签种类，i 指第 i 个标签。

GINI 系数越小，优先选择作为决策根节点

举个例子

数据	长鼻子(x)	大耳朵(y)	是否大象(1 or 0)
A1	1	1	1
A2	0	1	0
A3	1	0	0
A4	0	0	0

信息增益：g(D, A = “x”) = H(D)-H(D|x) = [-1/4*log(1/4) – 3/4*log(3/4)] （只有 1 个为大象）

– [**2/4***(-1/2*log(1/2) - 1/2*log(1/2) + **2/4***0)] = …

（2/4 为长鼻子，里面只有 1 个为大象，故为 1/2；非长鼻子中没有大象，故为 0）

信息增益率：gainRatio(D, A = “x”) = g(D, A = “x”) / $\text{SplitInfo}_A(D)$

= g(D, A = “x”) / (-1/2*log(1/2) - 1/2*log(1/2)

(1/2 为长鼻子)

Gini 指数： gini(D,A = “x”) = 2/4*[1-1/2*1/2] + 2/4*[1-0] = …

（2/4 的长鼻子中，1/2 为大象；2/4 的非长鼻子中，都不是大象，即 0）

其中，D 为标签，A 为属性。

三、连续数据的划分

为了适应决策树的特点，对于连续数据，我们需要将此化为离散的数据。

设定阈值 s，大于 s 为一类，小于 s 作为另一类，阈值的个数可为多个。以下列举两种方法：

1. 划分为多类

Given v values of A, then v-1 possible splits are evaluated. For example, the midpoint between the values a_i and a_{i+1} of A is $(a_i + a_{i+1})/2$

v 种值相邻值的中位数作为一个阈值，可划分为 v 类。

2. 划分为两类

取连续属性 A 中的最大值和最小值的中位数作为一个阈值，分成两类。

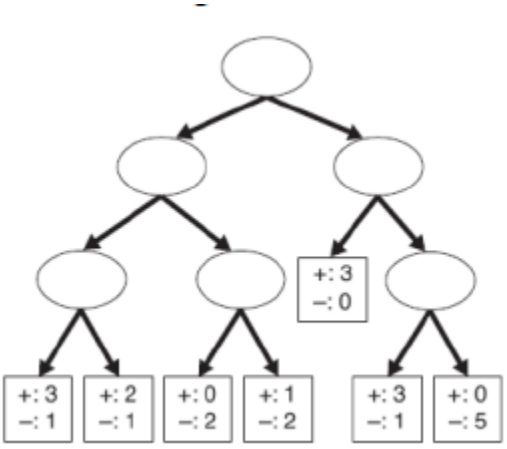
划分的方法很多，大多是人工设定。推荐根据预训练效果确定划分类数。

四、决策树的错误率

在一个决策树中，设分类错误的数据个数为 a，叶子结点个数为 b，一共有 n 个分类数据，假设一个惩罚值β。那么错误率的计算如下：

$$\varepsilon = \frac{a + b * \beta}{n}$$

例子：



错误率=（4+7*0.5）/24=0.3125

五、过拟合

加数据

数据量太小，远远不能代表现实中的总体，与现实总体相差太大，那么我们可以通过收集更多的数据来进行训练。

剪枝

标准的决策树中，每个属性都被“详细地”加以考虑，决策树的树叶节点所覆盖的训练样本会被描述的过于“细致”，即过度拟合训练集中的数据。剪枝是对决策树的一种简化，可以有效地避免过拟合。剪枝分为以下两类：

1. 前剪枝（Pre-pruning）

   建立某些规则限制决策树的充分生长。

   （1）错误率剪枝

   设定一个决策树的错误率阈值 s，根据阈值 s，决策当前的决策树要不要继续往下分，及早的停止树增长。

   （2）深度剪枝

   给定一个决策树深度的阈值 d，避免决策树过深。

2. 后剪枝（Post-pruning）

   待决策树充分生长完毕后再进行剪枝

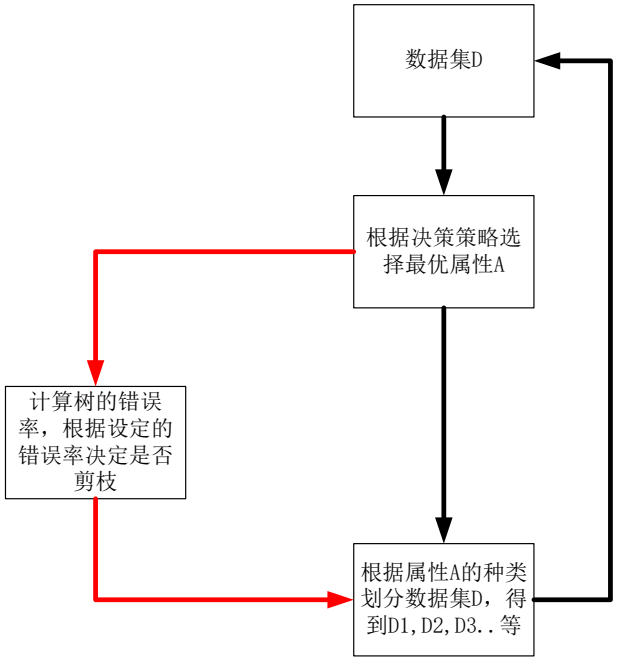
   （1） 错误率降低剪枝

   若发现当修剪后的树对于验证集的性能不会比原来的树差时，才真正删除该结点。

   实际上，无论前剪枝还是后剪枝，方法很多。如 Minimum Error Pruning(MEP)，Critical Value Pruning(CVP)，Optimal Pruning(OPP)，Cost–Sensitive Decision Tree Pruning(CSDTP)等方法，这些剪枝方法有不同的优化点，感兴趣的同学可以学习实践下。

六、实现举例

代码思路



根据这个代码流程框图，决策树的实现可以用递归的编程技巧实现。

python 实现举例

（相关链接：http://blog.csdn.net/alvine008/article/details/37760639，仅供参考）

```
1. def createDataSet():
2.     dataSet = [[1,1,'yes'],
3.                [1,1, 'yes'],
4.                [1,0,'no'],
5.                [0,1,'no'],
6.                [0,1,'no']]
7.     labels = ['no surfacing','flippers']
8.     return dataSet, labels
```

可以用 python 里面的 list 来存整个数据集，再本实例中存的形式如上。关于 list 数据结构，大家可以 google/百度一下“python list”，应有尽有。

```
1.  def createTree(dataSet, labels):
2.      classList = [example[-1] for example in dataSet]
3.      # the type is the same, so stop classify
4.      if classList.count(classList[0]) == len(classList):
5.          return classList[0]
6.      # traversal all the features and choose the most frequent feature
7.      if (len(dataSet[0]) == 1):
8.          return majorityCnt(classList)
9.      bestFeat = chooseBestFeatureToSplit(dataSet)
10.     bestFeatLabel = labels[bestFeat]
11.     myTree = {bestFeatLabel:{}}
12.     del(labels[bestFeat])
13.     #get the list which attain the whole properties
14.     featValues = [example[bestFeat] for example in dataSet]
15.     uniqueVals = set(featValues)
16.     for value in uniqueVals:
17.         subLabels = labels[:]
18.         myTree[bestFeatLabel][value] = createTree(splitDataSet(dataSet, bestFeat, value), subLabels)
19.     return myTree
```

1. python 巧妙地使用了 dict 来储存整个决策树（百度/google 搜索“python dict”），根据第 1 和第 18 行，createTree()是一个递归函数。
2. 关键的函数在于 chooseBestFeatureToSplit()，根据策略选择一个最优的属性/特征来划分数据集得到多个数据子集 D1, D2, D3，根据递归，对子集进行同样的处理即可。
3. 当没有属性/特征可以选择时候，也就是一条选择路径走尽的时候，根据多数投票的原则归类即可。
4. 需要实现的关键函数：
   i. chooseBestFeatureToSplit()-选择一个最优的特征划分数据集
   ii. splitDataSet()-根据第 i 步得到的最后特征划分数据

C++实现举例

实现思路(详看备注)：

参考数据结构

```
typedef vector<string> vs;
typedef vector<vs> vvs;
typedef vector<int> vi;
typedef map<string, int> msi;
typedef vector<double> vd;
```

```
struct node
{
    string splitOn;                 //分割属性名字
    string label;                   //标签
    bool isLeaf;                    //是否叶子
    vector<string> childrenValues;  //剩余属性
    vector<node*> children;         //子树
};
```

建树过程

```
node* buildDecisionTree(vvs &table, node* nodePtr, vvs &tableInfo)
{
    //没有待分割的值
    if (tableIsEmpty(table)) {
        return NULL;
    }
    //属性中所有值拥有相同标签
    if (isHomogeneous(table)) {
        nodePtr->isLeaf = true;
        nodePtr->label = table[1][table[1].size()-1];
        return nodePtr;
    } else {
        //得到待分割的属性 名字
        string splittingCol = decideSplittingColumn(table);
        nodePtr->splitOn = splittingCol;
        //返回这个属性的列号
        int colIndex = returnColumnIndex(splittingCol, tableInfo);
        int iii;
        for (iii = 1; iii < tableInfo[colIndex].size(); iii++) {
            node* newNode = (node*) new node;
            newNode->label = tableInfo[colIndex][iii];
            nodePtr->childrenValues.push_back(tableInfo[colIndex][iii]);
            newNode->isLeaf = false;
            newNode->splitOn = splittingCol;
            //基于要分割的属性，修建这个数据table，去除所有数据中的那一列属性元素
            vvs auxTable = pruneTable(table, splittingCol, tableInfo[colIndex][iii]);
            //递归建树
            nodePtr->children.push_back(buildDecisionTree(auxTable, newNode, tableInfo));
        }
    }
    return nodePtr;
}
```

（相关链接：http://blog.csdn.net/fy2462/article/details/31762429，仅供参考）

七、实验数据与评测

数据集

数据集采用`Breast Cancer Wisconsin (Diagnostic) Data Set'',乳腺癌诊断数据集。9 维特征向量，标签为 0（不患病） 1（患病）。一共包括两个文件：

train.csv: 583 个样例，每一个样例为 9 维特征向量+一个标签

test.csv: 100 个样例，每一个样例为 9 维特征向量+一个标签

实验任务

实现三种决策树模型（ID3, C4.5 以及 CART），用以上新闻数据集进行训练和测试。（注意特征数据需要离散化）

评测指标

1、准确度（Accuracy）

通过训练集训练后，在测试集实现分类预测，求出分类准确度。

2、决策树的错误率（ε）

计算训练出的最终决策树的错误率，惩罚值β默认设置为 0.5 和 1 两种，并解释β的意义。

3、展示关键代码，并解释。

4、决策树的改进方法，开阔自己的思路，改进决策树模型。

请将实验报告以及实验代码交到 FTP（ftp://my.ss.sysu.edu.cn/~ryh）