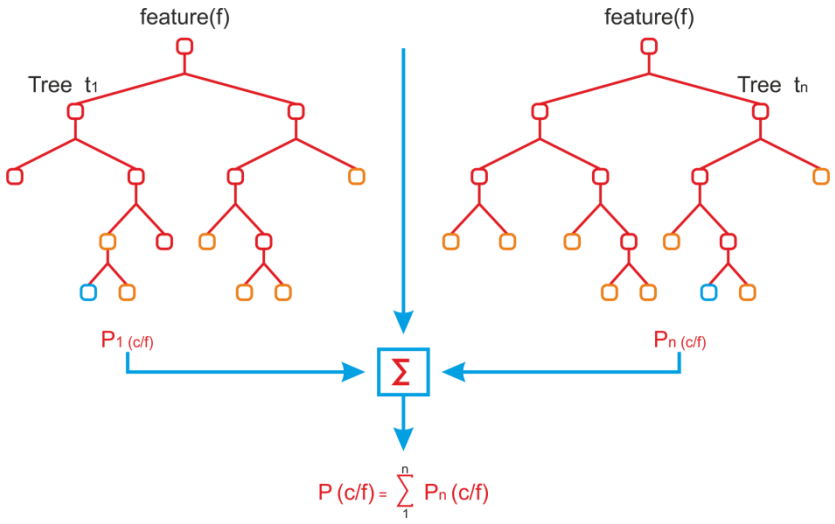
机器学习实验 14. 集成学习（二）随机森林

随机森林（Random Forest，简称 RF）原理概述

随机森林就是通过集成学习的思想将多棵树集成的一种算法，它的基本单元是决策树，因此本质上属于集成学习（Ensemble Learning）方法。所谓森林就是多个不连通的树构成的图，而树是最基本的图结构之一，在数据结构、离散数学等基础课程中早有论述。

如下图所示，随机森林就是根据多个树结构分别进行决策树分类，并将不同决策树的分类结果进行加权融合，获取最终的分类结果。



随机森林中树模型的生成

首先从决策树出发。

1）如果训练集大小为 N，对于每棵树而言，随机且有放回地从训练集中的抽取 N 个训练样本，作为该树的训练集。由此可知：每棵树的训练集都是不同的，而且里面包含重复的训练样本。

2）如果每个样本的特征维度为 M，指定一个常数 m<<M，随机地从 M 个特征中选取 m 个特征子集，每次树进行分裂时，从这 m 个特征中选择最优的来建

立树。

3）每棵树都尽最大程度的生长，并且没有剪枝过程。

4）将生成的多棵分类树组成随机森林，用随机森林分类器对新的数据进行判别与分类，分类结果按树分类器的投票多少而定。

- 为什么要随机抽样训练集?

这里随机包括数据采样随机和特征选择随机。如果不进行随机抽样，每棵树的训练集都一样，那么最终训练出的树分类结果也是完全一样的，这样的话完全没有 bagging 的必要。

- 为什么要有放回地抽样?

如果不是有放回的抽样，那么每棵树的训练样本都是不同的，都是交集相对来说比较小，这样每棵树都是"有偏的"，都是绝对"片面的"，也就是说每棵树训练出来都是有很大的差异的；而随机森林最后分类取决于多棵树（弱分类器）的投票表决，这种表决应该是"求同"，因此使用完全不同的训练集来训练每棵树这样对最终分类结果是没有帮助的，这样无异于是"盲人摸象"。

随机森林的特征选择功能

随机森林的一个重要用途是特征选择，通过尝试很多个决策树变量以检查变量在每棵树中表现的是最佳还是最糟糕。

例如，当第一次创建树模型时的特征用 fea1,fea2,fea3,和 fea4，得到错误率 error1。而第二次创建树时只用了 fea1、fea2 和 fea3，得到错误率 error2，比较 error1 和 error2，就可以知道特征 fea4 对模型训练的作用，是降低了模型的效果还是提升了其效果。所以需要计算出每一个特征的重要性并对这些特征进行一个排序，进而可以从所有特征中选择出重要性靠前的特征。sklearn.ensemble 的随机森林中为我们提供了 feature_importances_，可以调用的此方法来查看各个特征的重要性，实验部分将有展示。

随机森林算法在模拟数据中的一般实现

1) 导入数据集：

```
[1]: #随机森林
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score #交叉验证
import numpy as np
from sklearn.datasets import load_iris
iris=load_iris()  #从sklearn自带数据库中读取数据,iris以字典形式存储
    #iris的4个属性是：萼片宽度  萼片长度  花瓣宽度  花瓣长度
    #标签是花的种类：setosa versicolour virginica
X=iris['data']  #数据集大小为：150个数据样本，每个样本有四个特征
y=iris['target']  #数据对应的标签（有三类）:分别用 0,1,2 表示
print(X.shape)
print(y.shape)
```

```
(150, 4)
(150,)
```

2) 模型评测，并运用交叉验证的方法

```
[2]: rf=RandomForestClassifier(random_state=10,n_estimators=10)
scores=cross_val_score(rf,X,y,cv=5)
        #指定五次交叉验证
print(scores)
        #打印三次交叉验证的结果
print(scores.mean())
        #输出三次交叉验证的均值
```

```
[0.96666667 0.96666667 0.93333333 0.93333333 1.        ]
0.96
```

```
[3]: feat_name = iris["feature_names"] #读取样本的特征名
feat_name
```

```
t[3]: ['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

3) 运用随机森林来进行特征重要性的评估

- 读取特征名（共四个）


```
[3]: feat_name = iris["feature_names"] #读取样本的特征名
feat_name
```

```
ut[3]: ['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```


- 查看特征重要性

[4]:
```
rf.fit(X,y)
importance=rf.feature_importances_
    #通过调用feature_importances_得到每个特征的重要性
    #0—1之间，越大表示这个特征对模型学习越重要
imp_result=np.argsort(-importance)
    #通过np.argsort将importance值排序，默认是从小到大排序
    #这里用-importance，使其按从大到小顺序排
    #返回 imp_result 是index 数组
print(importance)
print(imp_result) #[3 2 0 1]表示importance[3]最大，importance[1]最小
```

```
[0.02554101 0.00964012 0.30135393 0.66346494]
[3 2 0 1]
```

- 按从小到大排列不同特征的重要性输出。

[5]:
```
for i in range(X.shape[1]):
    print("%2d. %-*s %f"%(i+1,30,feat_name[imp_result[i]],
                           importance[imp_result[i]]))
        #%-*s 代表输入一个字符串，-号代表左对齐、后补空白，
        #*号代表对齐宽度由输入时确定
```

```
 1. petal width (cm)               0.663465
 2. petal length (cm)              0.301354
 3. sepal length (cm)              0.025541
 4. sepal width (cm)               0.009640
```

4) 用 matplotlib 画图，直观展示各个特征的重要性

- 导入需要的库

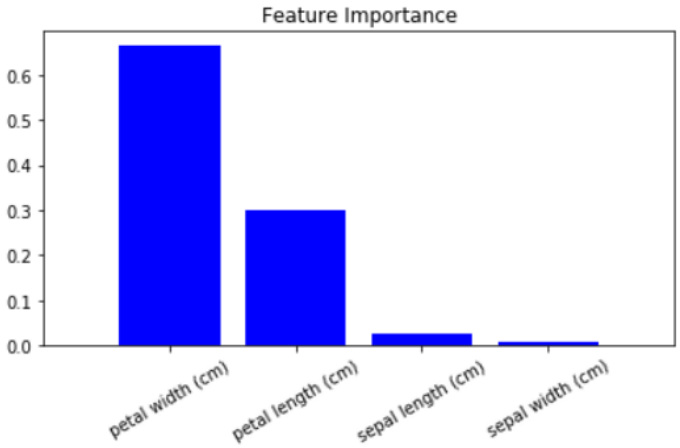
[6]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

- 将特征重要性按从大到小顺序排序，将样本名称与其重要性对应

[7]:
```
imp_sorted=[]
feat_name_sorted=[]
for i in range(X.shape[1]):
    imp_sorted.append(importance[imp_result[i]])
    feat_name_sorted.append(feat_name[imp_result[i]])
```

- 画图展示

[8]:
```
plt.figure()
plt.title('Feature Importance')
plt.bar(range(X.shape[1]),imp_sorted, color='blue', align='center')
    #参数分别是 样本横坐标、纵坐标，颜色，居中对齐
plt.xticks(range(X.shape[1]), feat_name_sorted, rotation=30)
plt.xlim([-1, X.shape[1]])
plt.tight_layout()
plt.show()
```


Feature Importance
0.6
0.5
0.4
0.3
0.2
0.1
0.0
petal width (cm)
petal length (cm)
sepal length (cm)
sepal width (cm)