第一部分 分类

第2章 K近邻算法

第3章 决策树

第4章 朴素贝叶斯

第5章 Logistic回归

第6章 支持向量机

第7章 AdaBoost

2.回归

第8章 回归

第9章 树回归

第9章介绍的是一种分类回归树CART(Classification And Regression Trees),该算法可以用于回归和分类。

* 9.1 复杂数据的局部建模

和第3章的决策树不同，决策树是一种贪心算法，需要在规定的时间内给出合适的决策，并不能考虑到全局最优。并且决策树使用的算法是ID3，每次选取当前的最佳特征去切割，特征使用后，在之后的划分过程中将不会再起作用，并且ID3不可以处理连续性的数据。CART可以通过二元切割来划分数据，并且稍作修改就可以处理回归问题。

* 9.2 连续和离散型特征数的构建

首先构建数据类型，新建一个字典，里面包含如下的数据：待切分的特征；待切分的特征值；右子树(不需要切分时是单值)；左子树。函数createTree()的伪代码如下：

```
找到最佳的待切分特征：
    如果该节点不能划分，将该节点存为叶节点
    执行二元划分
    在右子树调用createTree()
    在左子树调用createTree()
```

首先给出了三个函数 , , ,我们主要介绍函数 .

```
def createTree(dataSet,leafType=regLeaf,errType=regErr,ops=(1,4)):
    feat,val = chooseBestSplit(dataSet,leafType,errType,ops)
    if feat == None: return val
    # 创建一个空字典，'spInd'记录的当前的特征，'spVal'记录的是特征值val
    retTree()
    retTree['spInd'] = feat
    retTree['spVal'] = val
    #通过二分切割，得到两棵不同的子树
    lSet,rSet = binSplitDataSet(dataSet,feat,val)
    retTree['left'] = create(lSet,leafType,errType,ops)
    retTree['right'] = create(rSet,leafType,errType,ops)
    return retTree
```

* 9.3 将CART用于算法

* 9.3.1 构建数

函数运行，需要实现 函数，给定某个误差的计算方法，该函数会找到数据集的最佳二元切分方式。 只需要做两件事：切分数据集、生成相应的叶节点。其中， 是对创建叶节点的函数的引用，errType是对计算总方差函数的引用， 是用户自定义的参数。

```
def chooseBestSplit(dataSet,leafType=regLeaf,errType=regErr,ops=(1,4)):
    # tolS,tolN这两个参数用来控制函数的停止时机。tolS是容许误差的下降值，tolN是切分的最少样本数
    tolS = ops[0]; tolN = ops[1]
    # 如果数据集中所有的值相等，就推出
    if len(set(dataSet[:,-1].T.tolist()[0])) == 1:
        return None,leafType=(dataSet)
    m,n = shape(dataSet)
    # S是当前计算得到的总方差
    S = errType(dataSet)
    bestS = inf; bestIndex = 0; bestValue = 0
    for featIndex in range(n-1):
        for splitVal in set(dataSet[:,featIndex]):
            mat0,mat1 = binSplitDataSet(dataSet,featIndex,splitVal)
            # 如果当前的划分不满足条件，就跳过当前步，直接进行下一步
            if(shape(mat0)[0] < tolN) or (shape(mat1)[0] < tolN):continue
            if newS < bestS:
                bestIndex = featIndex
                beatValue = splitVal
                bestS = newS
    # 如果误差的减少的不够大，就直接退出
    if (S - bestS) < tolS:
        return None,leafType(dataSet)
    mat0,mat1 = binSplitDataSet(dataSet,bestIndex,bestValue)
    # 如果切分的数据集很小，就直接推出
    if (shape(mat0)[0] < tolN) or (shape(mat1)[0] < tolN):
        return None,leafType(dataSet)
    return bestIndex,bestValue
```

* 9.4 树剪枝

如果一棵树的节点过多，那么该模型可能会出现‘过拟合’现象，为了应对这种情况，需要对模型进行剪枝(pruning)，有两种剪枝方式，预剪枝(prepruning)和后剪枝(postpruning)，后剪枝需要使用测试集和训练集。

* 9.4.1 预剪枝

预剪枝就是在函数中提前结束对模型的划分，但是对参数　　较为敏感。

* 9.4.2 后剪枝

3.无监督学习

4.其他

10.27 begin:,,,