决策树算法

1. 决策树的构造

决策树的构造过程不依赖领域知识，它使用属性选择度量来选择将元组最好地划分成不同的类的属性。所谓决策树的构造就是进行属性选择度量确定各个特征属性之间的拓扑结构。

      构造决策树的关键步骤是分裂属性。所谓分裂属性就是在某个节点处按照某一特征属性的不同划分构造不同的分支，其目标是让各个分裂子集尽可能地“纯”。尽可能“纯”就是尽量让一个分裂子集中待分类项属于同一类别。分裂属性分为三种不同的情况：

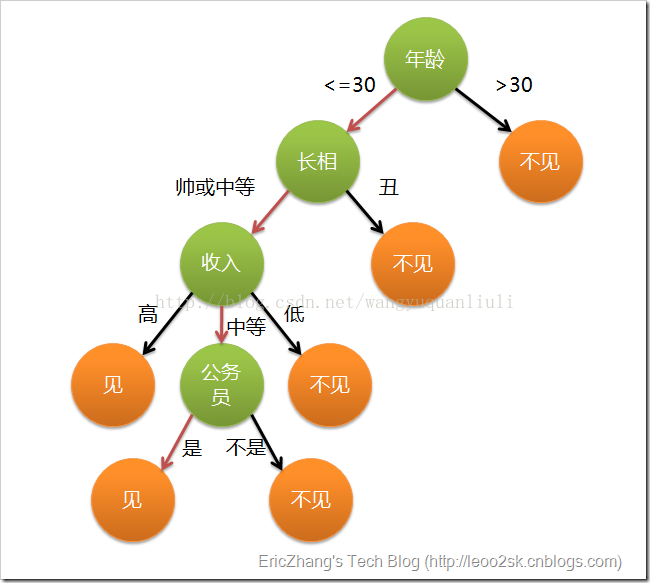
      1、属性是离散值且不要求生成二叉决策树。此时用属性的每一个划分作为一个分支。

      2、属性是离散值且要求生成二叉决策树。此时使用属性划分的一个子集进行测试，按照“属于此子集”和“不属于此子集”分成两个分支。

      3、属性是连续值。此时确定一个值作为分裂点split\_point，按照>split\_point和<=split\_point生成两个分支。

      构造决策树的关键性内容是进行属性选择度量，属性选择度量是一种选择分裂准则，是将给定的类标记的训练集合的数据划分D“最好”地分成个体类的[启发式方法](http://en.wikipedia.org/wiki/Heuristic_(computer_science))，它决定了拓扑结构及分裂点split\_point的选择。

1. 运用Id3算法解决以下问题



1. 构造数据集

**def** createDataSet():

    dataSet = [[1,1,'yes'],[1,1,'yes'],[1,0,'no'],[0,1,'no'],[0,1,'no']]

    features = ['no surfacing','flippers']

**return** dataSet,features

2、构造决策树

1. **def** treeGrowth(dataSet,features):
2. classList = [example[-1] **for** example **in** dataSet]
3. **if** classList.count(classList[0])==len(classList):
4. **return** classList[0]
5. **if** len(dataSet[0])==1:# no more features
6. **return** classify(classList)
8. bestFeat = findBestSplit(dataSet)#bestFeat is the index of best feature
9. bestFeatLabel = features[bestFeat]
10. myTree = {bestFeatLabel:{}}
11. featValues = [example[bestFeat] **for** example **in** dataSet]
12. uniqueFeatValues = set(featValues)
13. **del** (features[bestFeat])
14. **for** values **in** uniqueFeatValues:
15. subDataSet = splitDataSet(dataSet,bestFeat,values)
16. myTree[bestFeatLabel][values] = treeGrowth(subDataSet,features)
17. **return** myTree

3、

1. **def** classify(classList):
2. '''''
3. find the most in the set
4. '''
5. classCount = {}
6. **for** vote **in** classList:
7. **if** vote **not** **in** classCount.keys():
8. classCount[vote] = 0
9. classCount[vote] += 1
10. sortedClassCount = sorted(classCount.iteritems(),key = operator.itemgetter(1),reverse = True)
11. **return** sortedClassCount[0][0]

4、寻找用于分裂的最佳属性

1. **def** findBestSplit(dataset):
2. numFeatures = len(dataset[0])-1
3. baseEntropy = calcShannonEnt(dataset)
4. bestInfoGain = 0.0
5. bestFeat = -1
6. **for** i **in** range(numFeatures):
7. featValues = [example[i] **for** example **in** dataset]
8. uniqueFeatValues = set(featValues)
9. newEntropy = 0.0
10. **for** val **in** uniqueFeatValues:
11. subDataSet = splitDataSet(dataset,i,val)
12. prob = len(subDataSet)/float(len(dataset))
13. newEntropy += prob\*calcShannonEnt(subDataSet)
14. **if**(baseEntropy - newEntropy)>bestInfoGain:
15. bestInfoGain = baseEntropy - newEntropy
16. bestFeat = i
17. **return** bestFeat

5‘计算数据集的商’

1. **def** calcShannonEnt(dataset):
2. numEntries = len(dataset)
3. labelCounts = {}
4. **for** featVec **in** dataset:
5. currentLabel = featVec[-1]
6. **if** currentLabel **not** **in** labelCounts.keys():
7. labelCounts[currentLabel] = 0
8. labelCounts[currentLabel] += 1
9. shannonEnt = 0.0
11. **for** key **in** labelCounts:
12. prob = float(labelCounts[key])/numEntries
13. **if** prob != 0:
14. shannonEnt -= prob\*log(prob,2)
15. **return** shannonEnt

5进行数据的分类

1. **def** predict(tree,newObject):
2. **while** isinstance(tree,dict):
3. key = tree.keys()[0]
4. tree = tree[key][newObject[key]]
5. **return** tree
7. **if** \_\_name\_\_ == '\_\_main\_\_':
8. dataset,features = createDataSet()
9. tree = treeGrowth(dataset,features)
10. **print** tree
11. **print** predict(tree,{'no surfacing':1,'flippers':1})
12. **print** predict(tree,{'no surfacing':1,'flippers':0})
13. **print** predict(tree,{'no surfacing':0,'flippers':1})
14. **print** predict(tree,{'no surfacing':0,'flippers':0})

主程序：

1. **import** numpy as np
2. **import** struct
3. **import** matplotlib.pyplot as plt
5. filename = 'train-images.idx3-ubyte'
6. binfile = open(filename,'rb')#以二进制方式打开
7. buf = binfile.read()
9. index = 0
10. magic, numImages, numRows, numColums = struct.unpack\_from('>IIII',buf,index)#读取4个32 int
11. **print** magic,' ',numImages,' ',numRows,' ',numColums
12. index += struct.calcsize('>IIII')

15. im = struct.unpack\_from('>784B',buf,index)#每张图是28\*28=784Byte,这里只显示第一张图
16. index += struct.calcsize('>784B' )
18. im = np.array(im)
19. im = im.reshape(28,28)
20. **print** im
22. fig = plt.figure()
23. plt.imshow(im,cmap = 'binary')#黑白显示
24. plt.show()