Decision Tree Classifier

1.1 Introduction

A decision tree is a map of the possible outcomes of a series of related choices. It allows an individual or organization to weigh possible actions against one another based on their costs, probabilities, and benefits. They can be used either to drive informal discussion or to map out an algorithm that predicts the best choice mathematically.

A decision tree typically starts with a single node, which branches into possible outcomes. Each of those outcomes leads to additional nodes, which branch off into other possibilities. This gives it a treelike shape.

1.2 Building Decision Tree

We use Hunt’s algorithm for building decision tree. In this, a decision tree is grown in a recursive fashion by partitioning the training records into purer subset.

Let Dt be the set of training records that are associated with node t and y = {y1,y2,..,yc} be the class labels. The following is a recursive definition of Hunt’s Algorithm.

Step 1:

If all the records in Dt belong to the same class yt, then t is a leaf node labelled as yt.

Step 2:

If Dt contains records that belong to more than one class, an attribute test condition is selected to partition the records into smaller subsets. A child node is created for each outcome of the test condition and the records in Dt are distributed to the children based on the outcomes. The algorithm is then recursively applied to each child node.

Additional conditions to handle following cases:

1. If child node created in step 2 is empty, then node is declared leaf node with class label as majority of class of training records associated with its parent node.

2. If all records associated with a particular node have identical attribute value (except the class label), then the node is declared leaf node with class label as majority of records associated with the node have.

1.3 Selection of best split

Selection of the best split is done on the basis of degree of impurity of child nodes.

The impurity of a node measures how dissimilar the class labels are for the data instances belonging to a common node. Following are examples of measures that can be used to evaluate the impurity of a node t:

c-1

Entropy = −∑ pi(t) log2pi(t)

i=0

c-1

Gini index = 1 − ∑ pi(t)2

i=0

where pi(t) is the relative frequency of training instances that belong to class i at node t, c is the total number of classes, and 0 log20 = 0 in entropy calculations. Both measures give a zero impurity value if a node contains instances from a single class and maximum impurity if the node has equal proportion of instances from multiple classes.

1.4 Advantages

* Decision trees require relatively little effort from users for data preparation.
* Nonlinear relationships between parameters do not affect tree performance.

1.5 Disadvantages

* Preparing decision trees, especially large ones with many branches, are complex and time-consuming affairs.
* Decision trees are also prone to errors in classification, owing to differences in perceptions and the limitations of applying statistical tools.

2.1 Code for Implementation of Decision Tree Induction :

file=open("ver\_data.txt")

s=file.readlines()

file.close()

in\_data=[]

for rec in s:

in\_data.append(rec.rstrip('\n').split('\t'))

attri\_no=len(in\_data[0])-1

attr\_list=[1 for i in range(attri\_no)]

class node:

lable='not defined'

temp\_lable=''

test\_condition\_num=0

data=[]

child={}

def distribute(data,i):

mp={}

for record in data:

if(record[i] in mp):

mp[record[i]].append(record)

else:

mp[record[i]]=[record]

l=list(mp.values())

return l

def find\_gini(data):

total\_rec=len(data)

mp={}

for record in data:

if(record[-1] in mp):

mp[record[-1]]=mp[record[-1]]+1

else:

mp[record[-1]]=1

class\_count=list(mp.values())

gini=0;

for i in class\_count:

gini=gini+(i/total\_rec)\*\*2

gini=1-gini

return gini

def find\_test\_condition(data):

total\_rec\_parent=len(data)

best\_test\_condition=0

gini\_par=find\_gini(data)

max\_gain=0

for i in range(attri\_no):

if(attr\_list[i]==1):

l=distribute(data,i)

wt\_gini\_dist=0

for d in l:

wt\_gini\_dist=wt\_gini\_dist+(len(d)/total\_rec\_parent)\*find\_gini(d)

gain=gini\_par-wt\_gini\_dist

if(gain>max\_gain):

max\_gain=gain

best\_test\_condition=i

if(max\_gain==0):

return -1

elif(max\_gain>0):

return best\_test\_condition

else:

return -2

def build(root):

temp=root.data[0][-1]

flag=True

for record in root.data:

if(record[-1]!=temp):

flag=False

break

if(flag):

root.lable=temp

return

mp={}

for record in root.data:

if(record[-1] in mp):

mp[record[-1]]=mp[record[-1]]+1

else:

mp[record[-1]]=1

temp\_lable=list(mp.values())

temp\_lable=temp\_lable.index(max(temp\_lable))

temp\_lable=list(mp.keys())[temp\_lable]

root.temp\_lable=temp\_lable

root.test\_condition\_num=find\_test\_condition(root.data)

if(root.test\_condition\_num==-1):

root.lable=temp\_lable

return

elif(root.test\_condition\_num==-2):

print("getting -ve gain on the node")

root.lable=temp\_lable

return

attr\_list[root.test\_condition\_num]=0

child\_data=distribute(root.data,root.test\_condition\_num)

for i in child\_data:

root.child[i[0][root.test\_condition\_num]]=node()

root.child[i[0][root.test\_condition\_num]].data=i

k=list(root.child.keys())

for c in k:

build(root.child[c])

def classify(record,root):

if(root.lable!="not defined"):

return root.lable

t\_c\_n=root.test\_condition\_num

t\_c\_val=record[t\_c\_n]

if(t\_c\_val in root.child):

return classify(record,root.child[t\_c\_val])

else:

return root.temp\_lable

root=node()

root.data=in\_data

print("starting the build")

build(root)

print("build finished")

while(True) :

print()

print("enter record to classify or press 'x' to exit:")

print("bt skin birth aqua arial legs hibernation")

record=input()

if(record=='x'):

break

record=record.strip().split(' ')

if(len(record)!=attri\_no):

print("invalid record entered")

continue

class\_lable=classify(record,root)

print("class lable: ")

print(class\_lable)

2.2 TRAINING DATA SET (RECORD) FOR DECISION TREE ALGORITHM :

wb hair y n n y n mammal

cb scales n n n n y reptile

cb scales n y n n n fish

wb hair y y n n n mammal

cb none n semi n y y amphibian

cb scales n n n y n reptile

wb hair y n y y y mammal

wb feathers n n y y n bird

wb fur y n n y n mammal

cb scales y y n n n fish

cb scales n semi n y n reptile

wb feathers n semi n y n bird

wb quills y n n y y mammal

cb scales n y n n n fish

cb none n semi n y y amphibian