NAVEENA K N

USN: 1BM20CS411

MACHINE LEARNING

Lab Programs

1) Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples.

```
import csv
def updatehypothesis(x,h):
if h == []:
for i in range(0,len(x)):
h.append("$")
print("Initial State : ", h)
return x
for i in range(0,len(x)):
if x[i].upper() != h[i].upper() :
h[i] = "?"
print("Most Specific Hypothesis : ", h)
return h
if __name__ == "__main__":
data = []
h = []
with open('datasheet1.csv','r') as file :
reader = csv.reader(file)
print("Data Set : ")
for row in reader:
data.append(row)
print(row)
if data:
for x in data:
if x[-1].upper() == "YES":
x.pop()
```

```
    h = updatehypothesis(x,h)
print("Maximally Specific Hypothesis : " , h)
```

Output:

```
['sky', 'airtemp', 'humidity', 'wind', 'water', 'forecast', 'enjoysport']
['sunny', 'warm', 'normal', 'strong', 'warm', 'same', 'yes']
['sunny', 'warm', 'high', 'strong', 'warm', 'same', 'yes']
['rainy', 'cold', 'high', 'strong', 'warm', 'change', 'no']
['sunny', 'warm', 'high', 'strong', 'cool', 'change', 'yes']
Initial State :  ['$', '$', '$', '$', '$', '$']
Most Specific Hypothesis :  ['sunny', 'warm', '?', 'strong', 'warm', 'same']
Most Specific Hypothesis :  ['sunny', 'warm', '?', 'strong', '?', '?']
Maximally Specific Hypothesis :  ['sunny', 'warm', '?', 'strong', '?', '?']
```

2) For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.

```
import csv
a=[]
with open('datasheet1.csv','r') as csvfile:
for row in csv.reader(csvfile):
a.append(row)
print(a)
print("")
num_attributes = len(a[0])-1
s=['0']*num_attributes
g=[["?" for i in range(len(s))] for i in range(len(s))]
for i in range(0,len(a)):
if a[i][num_attributes]=='yes':
for j in range(0,num_attributes):
if s[j]=='0' or s[j]==a[i][j]:
s[j]=a[i][j]
else:
s[j]='?'
else:
for j in range(0,num_attributes):
if(s[j] == a[i][j] or s[j] =='?'):
g[j][j]='?'
continue
else:
g[j][j] = s[j]
for j in range(0,num_attributes):
if s[j]!=g[j][j] or s[j]=='?':
g[j][j]='?'
```

```
indices = [i for i, val in enumerate(g) if val == ['?', '?', '?', '?', '?', '?']]
for i in indices:
g.remove(['?', '?', '?', '?', '?', '?'])
print("Specific hypothesis:",s)
print("General hypothesis:",g)
```

Output:

```
[['sky', 'airtemp', 'humidity', 'wind', 'water', 'forecast', 'enjoysport'],
['sunny', 'warm', 'normal', 'strong', 'warm', 'same', 'yes'],
['sunny', 'warm', 'high', 'strong', 'warm', 'same', 'yes'],
['rainy', 'cold', 'high', 'strong', 'warm', 'change', 'no'],
['sunny', 'warm', 'high', 'strong', 'cool', 'change', 'yes']]

Specific hypothesis: ['sunny', 'warm', '?', 'strong', '?', '?']
General hypothesis: [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?']]
```

3) Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

```
import math
import csv
def load_csv(id3):
lines=csv.reader(open('../input/mlid3/id3.csv',"r"));
dataset = list(lines)
headers = dataset.pop(0)
return dataset,headers
class Node:
def __init__(self,attribute):
self.attribute=attribute
self.children=[]
self.answer=""
def subtables(data,col,delete):
dic={}
coldata=[row[col] for row in data]
attr=list(set(coldata))
counts=[0]*len(attr)
r=len(data)
c=len(data[0])
for x in range(len(attr)):
for y in range(r):
if data[y][col]==attr[x]:
counts[x]+=1
for x in range(len(attr)):
dic[attr[x]]=[[0 for i in range(c)] for j in range(counts[x])]
pos=0
for y in range(r):
if data[y][col]==attr[x]:
```

```
if delete:
del data[y][col]
dic[attr[x]][pos]=data[y]
pos+=1
return attr,dic
def entropy(S):
attr=list(set(S))
if len(attr)==1:
return 0
counts=[0,0]
for i in range(2):
counts[i]=sum([1 for x in S if attr[i]==x])/(len(S)*1.0)
sums=0
for cnt in counts:
sums+=-1*cnt*math.log(cnt,2)
return sums
def compute_gain(data,col):
attr,dic = subtables(data,col,delete=False)
total_size=len(data)
entropies=[0]*len(attr)
ratio=[0]*len(attr)

total_entropy=entropy([row[-1] for row in data])
for x in range(len(attr)):
ratio[x]=len(dic[attr[x]])/(total_size*1.0)
entropies[x]=entropy([row[-1] for row in dic[attr[x]]])
total_entropy-=ratio[x]*entropies[x]
return total_entropy
def build_tree(data,features):
lastcol=[row[-1] for row in data]
```

```
if(len(set(lastcol)))==1:
node=Node("")
node.answer=lastcol[0]
return node
n=len(data[0])-1
gains=[0]*n
for col in range(n):
gains[col]=compute_gain(data,col)
split=gains.index(max(gains))
node=Node(features[split])
fea = features[:split]+features[split+1:]
attr,dic=subtables(data,split,delete=True)
for x in range(len(attr)):
child=build_tree(dic[attr[x]],fea)
node.children.append((attr[x],child))
return node
def print_tree(node,level):
if node.answer!="":
print("  "*level,node.answer)
return
print("  "*level,node.attribute)
for value,n in node.children:
print("  "*(level+1),value)
print_tree(n,level+2)
def classify(node,x_test,features):
if node.answer!="":
print(node.answer)
return
pos=features.index(node.attribute)
for value, n in node.children:
```

```
 if x_test[pos]==value:
classify(n,x_test,features)
dataset,features=load_csv("../input/mlid3/id3.csv")
node1=build_tree(dataset,features)
print("The decision tree for the dataset using ID3 algorithm is")
print_tree(node1,0)
# testdata,features=load_csv("../input/playtennis/ID3.csv")
# print(features,"\n\n",testdata)
# for xtest in testdata:
#     print("The test instance:",xtest)
#     print("The label for test instance:",end="   ")
#     classify(node1,xtest,features)
```

Dataset:

1	Outlook	Temperature	Humidity	Wind	Answer
2	sunny	hot	high	weak	no
3	sunny	hot	high	strong	no
4	overcast	hot	high	weak	yes
5	rain	mild	high	weak	yes
6	rain	cool	normal	weak	yes
7	rain	cool	normal	strong	no
8	overcast	cool	normal	strong	yes
9	sunny	mild	high	weak	no
10	sunny	cool	normal	weak	yes
11	rain	mild	normal	weak	yes
12	sunny	mild	normal	strong	yes
13	overcast	mild	high	strong	yes
14	overcast	hot	normal	weak	yes
15	rain	mild	high	strong	no

Output:

```
The decision tree for the dataset using ID3 algorithm is
 Outlook
   sunny
     Humidity
       normal
         yes
       high
         no
   overcast
     yes
   rain
     Wind
       strong
         no
       weak
         yes
```

4) Write a program to implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.

```
import pandas as pd
data = pd.read_csv('../input/playtennisnb/PlayTennis.csv')
data.head()
y = list(data['PlayTennis'].values)
X = data.iloc[:,1:].values
print(f'Target Values: {y}')
print(f'Features: \n{X}')
y_train = y[:8]
y_val = y[8:]
X_train = X[:8]
X_val = X[8:]
print(f"Number of instances in training set: {len(X_train)}")
print(f"Number of instances in testing set: {len(X_val)}")
class NaiveBayesClassifier:
def __init__(self, X, y):
self.X, self.y = X, y
self.N = len(self.X)
self.dim = len(self.X[0])
self.attrs = [[] for _ in range(self.dim)]
self.output_dom = {}
self.data = []
for i in range(len(self.X)):
for j in range(self.dim):
if not self.X[i][j] in self.attrs[j]:
self.attrs[j].append(self.X[i][j])
if not self.y[i] in self.output_dom.keys():
self.output_dom[self.y[i]] = 1
else:
self.output_dom[self.y[i]] += 1
```

```
self.data.append([self.X[i], self.y[i]])
def classify(self, entry):
solve = None
max_arg = -1
for y in self.output_dom.keys():
prob = self.output_dom[y]/self.N
for i in range(self.dim):
cases = [x for x in self.data if x[0][i] == entry[i] and x[1] == y]
n = len(cases)
prob *= n/self.N
if prob > max_arg:
max_arg = prob
solve = y
return solve
nbc = NaiveBayesClassifier(X_train, y_train)
total_cases = len(y_val)
good = 0
bad = 0
predictions = []
for i in range(total_cases):
predict = nbc.classify(X_val[i])
predictions.append(predict)
if y_val[i] == predict:
good += 1
else:
bad += 1
print('Predicted values:', predictions)
print('Actual values:', y_val)
print()
print('Total number of testing instances in the dataset:', total_cases)
print('Number of correct predictions:', good)
print('Number of wrong predictions:', bad)
```

```
print()

print('Accuracy of Bayes Classifier:', good/total_cases)
```

Dataset:

1		PlayTennis	Outlook	Temperature	Humidity	Wind
2		No	Sunny	Hot	High	Weak
3		No	Sunny	Hot	High	Strong
4		Yes	Overcast	Hot	High	Weak
5		Yes	Rain	Mild	High	Weak
6		Yes	Rain	Cool	Normal	Weak
7		No	Rain	Cool	Normal	Strong
8		Yes	Overcast	Cool	Normal	Strong
9		No	Sunny	Mild	High	Weak
10		Yes	Sunny	Cool	Normal	Weak
11		Yes	Rain	Mild	Normal	Weak
12		Yes	Sunny	Mild	Normal	Strong
13		Yes	Overcast	Mild	High	Strong
14		Yes	Overcast	Hot	Normal	Weak
15		No	Rain	Mild	High	Strong

Output:

```
Enter some code at the bottom of this console and press [Enter].
  PlayTennis   Outlook Temperature Humidity    Wind
0         No     Sunny         Hot     High    Weak
1         No     Sunny         Hot     High  Strong
2        Yes  Overcast         Hot     High    Weak
3        Yes      Rain        Mild     High    Weak
4        Yes      Rain        Cool   Normal    Weak
Target Values: ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']
Features:
[['Sunny' 'Hot' 'High' 'Weak']
 ['Sunny' 'Hot' 'High' 'Strong']
 ['Overcast' 'Hot' 'High' 'Weak']
 ['Rain' 'Mild' 'High' 'Weak']
 ['Rain' 'Cool' 'Normal' 'Weak']
 ['Rain' 'Cool' 'Normal' 'Strong']
 ['Overcast' 'Cool' 'Normal' 'Strong']
 ['Sunny' 'Mild' 'High' 'Weak']
 ['Sunny' 'Cool' 'Normal' 'Weak']
 ['Rain' 'Mild' 'Normal' 'Weak']
 ['Sunny' 'Mild' 'Normal' 'Strong']
 ['Overcast' 'Mild' 'High' 'Strong']
 ['Overcast' 'Hot' 'Normal' 'Weak']
 ['Rain' 'Mild' 'High' 'Strong']]
Number of instances in training set: 8
Number of instances in testing set: 6
Predicted values: ['No', 'Yes', 'No', 'Yes', 'Yes', 'No']
Actual values: ['Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']

Total number of testing instances in the dataset: 6
Number of correct predictions: 4
Number of wrong predictions: 2

Accuracy of Bayes Classifier: 0.6666666666666666
```

5) Implement the Linear Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('../input/salarydata/salaryData.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3, random_state=0)
# Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
# Predicting the Test set results
y_pred = regressor.predict(X_test)
# Visualizing the Training set results
viz_train = plt
viz_train.scatter(X_train, y_train, color='green')
viz_train.plot(X_train, regressor.predict(X_train), color='black')
viz_train.title('Salary VS Experience (Training set)')
viz_train.xlabel('Year of Experience')
viz_train.ylabel('Salary')
viz_train.show()
# Visualizing the Test set results
viz_test = plt
viz_test.scatter(X_test, y_test, color='green')
viz_test.plot(X_train, regressor.predict(X_train), color='black')
viz_test.title('Salary VS Experience (Test set)')
viz_test.xlabel('Year of Experience')
viz_test.ylabel('Salary')
viz_test.show()
```

Dataset:

1	YearsExperience	Salary
2	1.1	39343
3	1.3	46205
4	1.5	37731
5	2.0	43525
6	2.2	39891
7	2.9	56642
8	3.0	60150
9	3.2	54445
10	3.2	64445
11	3.7	57189
12	3.9	63218
13	4.0	55794
14	4.0	56957
15	4.1	57081
16	4.5	61111
17	4.9	67938
18	5.1	66029
19	5.3	83088
20	5.9	81363
21	6.0	93940
22	6.8	91738
23	7.1	98273
24	7.9	101302
25	8.2	113812
26	8.7	109431
27	9.0	105582
28	9.5	116969
29	9.6	112635
30	10.3	122391
31	10.5	121872

Output:


Salary VS Experience (Training set)
Salary
Year of Experience
120000
100000
80000
60000
40000
2
4
6
8
10