Date 06/10/2020

Expt. No. 11 Page No. 05

11. 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 hypothesis consistent with the training examples.

```
import csv
with open ('EnjoySport.csv') as csvFile:
    examples = [tuple (line) for line in csv.reader (csvFile)]
print (examples)

def get_domains (examples):
    d = [set() for i in examples[0]]
    for x in examples:
        for i, xi in enumerate(x):
            d[i].add (xi)
    return [list[sorted(x)) for x in d]

get_domains(examples)

def g_0(n):
    return ('?', )*n

def s_0(n):
    return ('0', )*n

def more_general (h1, h2):
    more_general_parts=[]
    for x, y in zip (h1, h2):
        mg = x == '?' or (x != '0' and (x==y or y == '0'))
```

Teacher's Signature : _____

```
        more_general_parts.append(mg)
    return all(more_general_parts)

def consistent(hypothesis, example):
    return more_general(hypothesis, example)

def min_generalizations(h,x):
    h_new = list(h)
    for i in range(len(h)):
        if not consistent(h[i:i+1], x[i:i+1]):
            if h[i] != 'o':
                h_new[i]='?'
            else:
                h_new[i]=x[i]
    return [tuple(h_new)]

def generalize_S(x,G,S):
    S_prev = list(S)
    for s in S_prev:
        if s not in S:
            continue
        if not consistent(S,x):
            S.remove(S)
            Splus= min_generalizations(S,x)
            S.update([h for h in S plus if any([more_general(g,h)
                                    for g in G])])
            S.difference_update([h for h in S if any([
                                    more_general(h,h1)
```

Teacher's Signature :

```
    for h1 in S if h!=h1 ])])
return S

def min_Specializations (h, domains, x):
    results =[]
    for i in range (len(h)):
        if h[i] == '?':
            for val in domains[i]:
                if x[i] != val:
                    h_new = h[:i] +(val,) + h[i+1:]
                    results.append(h_new)
        elif  elif h[i]!='0':
                h_new = h[:i] + ('0',) + h[i+1:]
                results.append(h_new)
    return results

def Specialize_G(x, domains, G, S):
    G_prev = list(G)
    for g in G_prev:
        if g not in G:
            continue
        if consistent(g,x):
            G.remove(g)
            Gminus = min_specializations(g, domains, x)
            G.update([h for h in G minus if any ([
                more_general(h,s) for s in S])])
            G.difference_update([h for h in G if any
                ([more_general(g1,h) for g1 in G if h!=g1])])
    return G
```

Teacher's Signature :

Date 06/10/2020

Expt. No. 11

Page No. 28

```
def candidate_elimination(examples):
    domains = get_domains(examples)[:-1]
    G = set([tuple(['?' for i in range(len(domains))])])
    S = set([tuple(['0' for i in range(len(domains))])])
    i=0
    print('All the hypotheses in General & Specific
          boundary are:\n')
    print('\n G[{0}]:'.format(i), G)
    print('\n S[{0}]:'.format(i), S)
    for xcx in examples:
        i = i+1
        x, cx = xcx[:-1], xcx[-1]
        if cx=='Yes':
            G = {g for g in G if consistent(g,x)}
            S = generalize_S(x,G,S)
        else:
            S = {s for s in S if not consistent(s,x)}
            G = specialize_G(x, domains, G, S)
        print('\n G[{0}]:', format(i), G)
    print('\n S[{0}]:', format(i), S)
    return

Candidate_elimination(example)
```

Dataset:

Japan	Honda	Blue	1980	Economy	Yes
Japan	Toyota	Green	1970	Sports	No
Japan	Toyota	Blue	1990	Economy	Yes
USA	Chrysler	Red	1980	Economy	No
Japan	Honda	White	1980	Economy	Yes

Output:

```
[('Japan', 'Honda', 'Blue', '1980', 'Economy', 'Yes'),
 ('Japan', 'Toyota', 'Green', '1970', 'Sports', 'No'),
 ('Japan', 'Toyota', 'Blue', '1990', 'Economy', 'Yes'),
 ('USA', 'Chrysler', 'Red', '1980', 'Economy', 'No'),
 ('Japan', 'Honda', 'White', '1980', 'Economy', 'Yes')]

[['Japan', 'USA'],
 ['Chrysler', 'Honda', 'Toyota'],
 ['Blue', 'Green', 'Red', 'White'],
 ['1970', '1980', '1990'],
 ['Economy', 'Sports'],
 ['No', 'Yes']]
```

All the hypothesis in General and Specific boundary are:

G[0]: { ('?', '?', '?', '?', '?')}

S[0]: { ('0','0','0','0','0')}

G[1]: { ('?','?','?','?','?')}

S[1]: { ('Japan', 'Honda', 'Blue', 1980, 'Economy')}

G[2]: { ('?','?','?','?','Economy'),
('?','Honda','?','?','?'),
('?','?','?','1980','?'),
('?','?','Blue','?','?')}

S[2]: { ('Japan','Honda','Blue','1980','Economy')}

G[3]: { ('?','?','?','?','Economy'),
('?','?','Blue','?','?')}

S[3]: { ('Japan','?','Blue','?','Economy')}

G[4]: { ('Japan','?','?','?','Economy'), ('?','?','Blue','?','?')}

S[4]: { ('Japan','?','Blue','?','Economy')}

G[5]: { ('Japan','?','?','?','Economy')}

S[5]: { ('Japan','?','?','?','Economy')}