# Candidate Elimination Algorithm

from google.colab import drive

drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force\_remount=True).

import csv

with open('/content/drive/My Drive/ML LAB/EconomyCar.csv') as csvFile:

examples = [tuple(line) for line in csv.reader(csvFile)]

print(examples)

[('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')]

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)

[['Japan ', 'USA'],

['Chrysler', 'Honda', 'Toyota'],

['Blue ', 'Green', 'Red', 'White'],

['1970', '1980', '1990'],

['Economy', 'Sports'],

['No', 'Yes']]

def g\_0(n):

return ('?',)\*n

def s\_0(n):

return ('Phi',)\*n

def more\_general(h1, h2):

more\_general\_parts = []

for x, y in zip(h1, h2):

mg = x == '?' or (x != 'Phi' and (x == y or y == 'Phi'))

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] != 'Phi':

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 Splus if any([more\_general(g,h)

for g in G])])

S.difference\_update([h for h in S if

any([more\_general(h, h1)

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 h[i] != 'Phi':

h\_new = h[:i] + ('Phi',) + 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 Gminus 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

def candidate\_elimination(examples):

domains = get\_domains(examples)[:-1]

G = set([g\_0(len(domains))])

S = set([s\_0(len(domains))])

i=0

print('All the hypotheses in General and 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(examples)

All the hypotheses in General and Specific boundary are:

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

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

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

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

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

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')}