**3.** **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.**

THEORY: The candidate elimination algorithm incrementally builds the version space given a hypothesis space H and a set E of examples. The examples are added one by one; each example possibly shrinks the version space by removing the hypotheses that are inconsistent with the example. The candidate elimination algorithm does this by updating the general and specific boundary for each new example.

* Concept learning: Concept learning is basically learning task of the machine (Learn by Train data)
* General Hypothesis: Not Specifying features to learn the machine.
* G = {‘?’, ‘?’,’?’,’?’…}: Number of attributes
* Specific Hypothesis: Specifying features to learn machine (Specific feature)
* S= {‘pi’,’pi’,’pi’…}: Number of pi depends on number of attributes.
* Version Space: It is intermediate of general hypothesis and Specific hypothesis. It not only just written one hypothesis but a set of all possible hypothesis based on training data-set.

**Algorithm:**

**Step1**: Load Data set

**Step2**: Initialize General Hypothesis and Specific Hypothesis.

**Step3**: For each training example

**Step4**: If example is positive example

if attribute\_value == hypothesis\_value:

Do nothing

else:

replace attribute value with '?' (Basically generalizing it)

**Step5**:If example is Negative example

Make generalize hypothesis more specific.

**PROCEDURE/PROGRAM:**

import csv

with open('cea1.csv') as csvFile:

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

print(examples)

[('sunny', 'warm', 'normal', 'strong', 'warm', 'same', '1'), ('sunny', 'warm', 'high', 'strong', 'warm', 'same', '1'), ('rainy', 'cold', 'high', 'strong', 'warm', 'change', '0'), ('sunny', 'warm', 'high', 'strong', 'cold', 'change', '1')]

**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)

[['rainy', 'sunny'],

['cold', 'warm'],

['high', 'normal'],

['strong'],

['cold', 'warm'],

['change', 'same'],

['0', '1']]

**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'))

        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] != '0':

                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] != '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 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=='1':

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

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

S[1]: {('sunny', 'warm', 'normal', 'strong', 'warm', 'same')}

G[2]: {('?', '?', '?', '?', '?', '?')}

S[2]: {('sunny', 'warm', '?', 'strong', 'warm', 'same')}

G[3]: {('?', '?', '?', '?', '?', 'same'), ('?', 'warm', '?', '?', '?', '?'), ('sunny', '?', '?', '?', '?', '?')}

S[3]: {('sunny', 'warm', '?', 'strong', 'warm', 'same')}

G[4]: {('?', 'warm', '?', '?', '?', '?'), ('sunny', '?', '?', '?', '?', '?')}