Candidate Elimination Algorithm

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.

You can consider this as an extended form of the Find-S algorithm.

Consider both positive and negative examples.

Actually, positive examples are used here as the Find-S algorithm (Basically they are generalizing from the specification).

While the negative example is specified in the generalizing form.

Terms Used:

Concept learning: Concept learning is basically the 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’…}: The number of pi depends on a number of attributes.

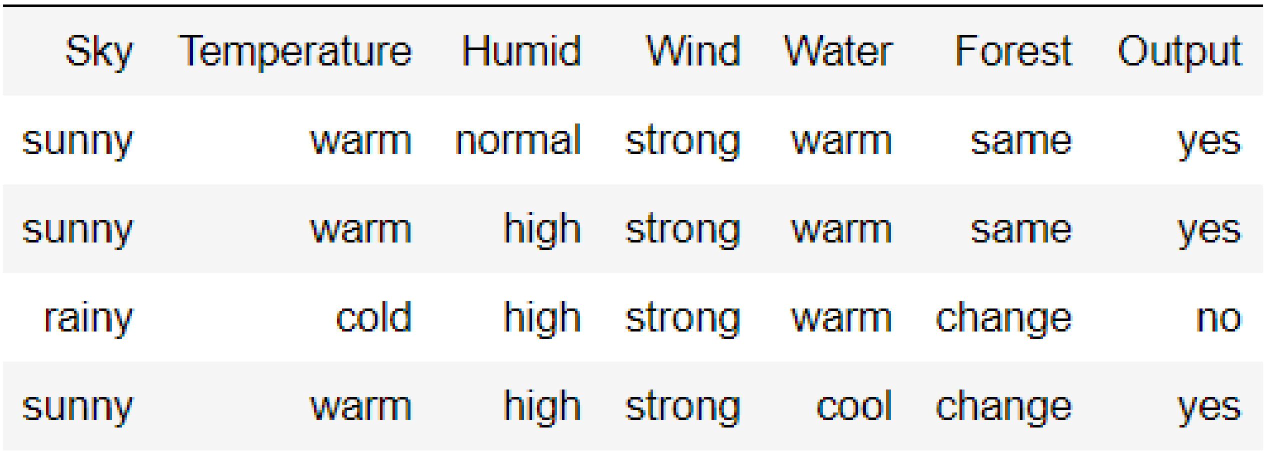
Version Space: It is an intermediate of general hypothesis and Specific hypothesis. It not only just writes one hypothesis but a set of all possible hypotheses based on training data-set.

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.
```

Example:

Consider the dataset given below:



Sky	Temperature	Humid	Wind	Water	Forest	Output
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

Algorithmic steps:

Initially : G = [[?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?],

[?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?]]

S = [Null, Null, Null, Null, Null, Null]

For instance 1 : <‘sunny','warm','normal','strong','warm ','same'> and positive output.

G1 = G

S1 = ['sunny','warm','normal','strong','warm ','same']

For instance 2 : <'sunny','warm','high','strong','warm ','same'> and positive output.

G2 = G

S2 = ['sunny','warm',?,'strong','warm ','same']

For instance 3 : <'rainy','cold','high','strong','warm ','change'> and negative output.

G3 = [['sunny', ?, ?, ?, ?, ?], [?, 'warm', ?, ?, ?, ?], [?, ?, ?, ?, ?, ?],

[?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, 'same']]

S3 = S2

For instance 4 : <'sunny','warm','high','strong','cool','change'> and positive output.

G4 = G3

S4 = ['sunny','warm',?,'strong', ?, ?]

At last, by synchronizing the G4 and S4 algorithm produce the output.

Output :

G = [['sunny', ?, ?, ?, ?, ?], [?, 'warm', ?, ?, ?, ?]]

S = ['sunny','warm',?,'strong', ?, ?]