# 1. Find S Algorithm

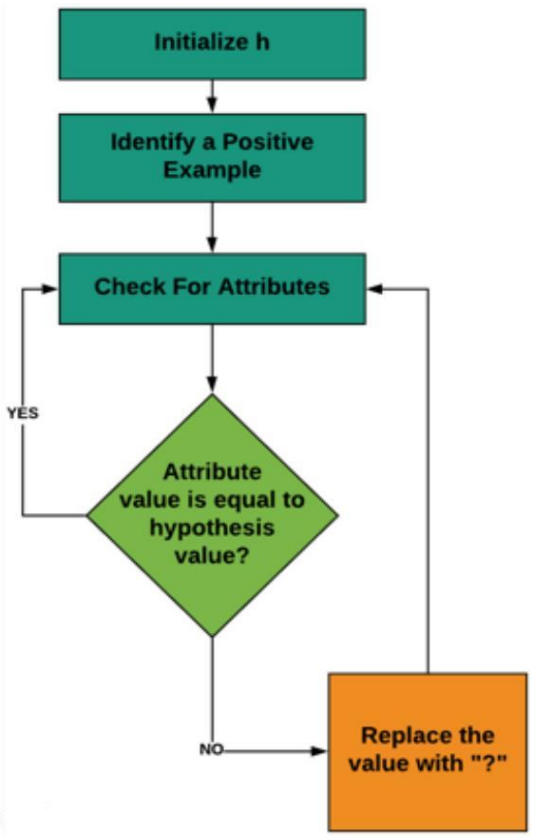
The find-S algorithm is a basic concept learning algorithm in machine learning. The find-S algorithm finds the most specific hypothesis that fits all the positive examples. We have to note here that the algorithm considers only those positive training example. The find-S algorithm starts with the most specific hypothesis and generalizes this hypothesis each time it fails to classify an observed positive training data. Hence, the Find-S algorithm moves from the most specific hypothesis to the most general hypothesis.

## Flow chart/Algorithm:

### Algorithm:

1. Initialize ‘h’ to the most specific hypothesis.
2. The Find-S algorithm only considers the positive examples and eliminates negative examples. For each positive example, the algorithm checks for each attribute in the example. If the attribute value is the same as the hypothesis value, the algorithm moves on without any changes. But if the attribute value is different than the hypothesis value, the algorithm changes it to ‘?’.

### Flow chart:



## Results and Discussion:

```
================= RESTART: E:\python\Python Codes\find-s.py =================
['Citations', 'Size', 'Library', 'Price', 'Editions', 'Buy']
['Some', 'Small', 'no', 'Affordable', 'One', 'No']
['Many', 'Big', 'no', 'Expensive', 'Many', 'Yes']
['Many', 'Medium', 'no', 'Expensive', 'Few', 'Yes']
['Many', 'Small', 'no', 'Affordable', 'Many', 'Yes']
**********************************************************
the maximally specific hypothesis is [['Many', '?', 'no', '?', '?']]
>>>


================= RESTART: E:\python\Python Codes\find-s.py =================
['Origin', 'Manufacturer', 'Color', 'Decade', 'Type', 'Example Type']
['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']
**********************************************************************
the maximally specific hypothesis is [['Japan', '?', '?', '?', 'Economy']]
>>>
```

Here from the results we can conclude that the data set which is stored in the csv file has been extracted and stored in a list of lists and the training examples are iterated after the hypothesis is initialized with specific hypothesis i.e. [“0”,“0”,“0”,“0”,“0”] and then the first true training example is assigned with the specific hypothesis ,and then the algorithm is executed as the true training examples are compared with the hypothesis and if they are not equal the attribute is generalized .Later after the final hypothesis is generated ,then the hypothesis is verified with a testing example ,which gives us the result based on the hypothesis weather the example is true or false.

# 2. Candidate Elimination Algorithm

The candidate elimination algorithm incrementally builds the version space given a hypothesis space H and a set D 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 Find-S algorithm.
- Consider both positive and negative examples.
- Actually, positive examples are used here as Find-S algorithm (Basically they are generalizing from the specification).
- While the negative example is specified from generalize form.

**Flow chart/Algorithm:**

**Algorithm:**

For each training example d, do:

If d is positive example

Remove from G any hypothesis h inconsistent with d

For each hypothesis s in S not consistent with d:

Remove s from S

Add to S all minimal generalizations of s consistent with d and having a generalization in G

Remove from S any hypothesis with a more specific h in S○

If d is negative example

Remove from S any hypothesis h inconsistent with d

For each hypothesis g in G not consistent with d:

Remove g from G

Add to G all minimal specializations of g consistent with d and having a specialization in S

Remove from G any hypothesis having a more general hypothesis in G

## Results and Discussion:

```
================ RESTART: E:\python\Python Codes\candidate.py ================

Steps of Candidate Elimination Algorithm 1
['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '
?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?
', '?']]

Steps of Candidate Elimination Algorithm 2
['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '
?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?
', '?']]

Steps of Candidate Elimination Algorithm 3
['Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '
?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?
', '?']]

Steps of Candidate Elimination Algorithm 4
['Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same']
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '
?', '?', 'Same']]

Steps of Candidate Elimination Algorithm 5
['Sunny', 'Warm', '?', 'Strong', '?', '?']
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '
?', '?', '?']]

Final specific hypothesis:
 ['Sunny', 'Warm', '?', 'Strong', '?', '?']

Final general hypothesis:
 [['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?']]
>>>
```

The each example is iterated through a for loop ,the true examples are iterated in a way that the specific hypothesis is generalized and the most general hypothesis is checked weather It classifies the example, the false examples are iterated in a way that the general hypothesis is made specific by adding the specific hypothesis to the general hypothesis set. Therefore ,we get the most specific hypothesis in the first iteration as it is true example and it classifies the true example ,the specific hypothesis is generalized in the same way as in find s and in third iteration as It is a false (negative)example it adds the specific hypothesis to the general hypothesis and the specific hypothesis is unchanged and in the fourth iteration the general hypothesis the is checked weather all the hypothesis In the set classify the example the non classifying hypothesis is removed,Gives us the final general and specific hypothesis which gives us the version space.

# 3. Artificial Neural Networks

The study of artificial neural networks (ANNs) has been inspired in part by the observation that biological learning systems are built of very complex webs of interconnected neurons in brains. Generally, ANNs are built out of a densely interconnected set of simple units, where each unit takes a number of real-valued inputs and produces a single real-valued output.

**PERCEPTRON**

One type of ANN system is based on a unit called a perceptron. Perceptron is a single layer neural network. Perceptron is a function that maps its input "x," which is multiplied with the learned weight coefficient; an output value "f(x)"is generated.

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

In the equation given above:

- "w" = vector of real-valued weights
- "b" = bias (an element that adjusts the boundary away from origin without any dependence on the input value)
- "x" = vector of input x values

$$\sum_{i=1}^{m} w_i x_i$$

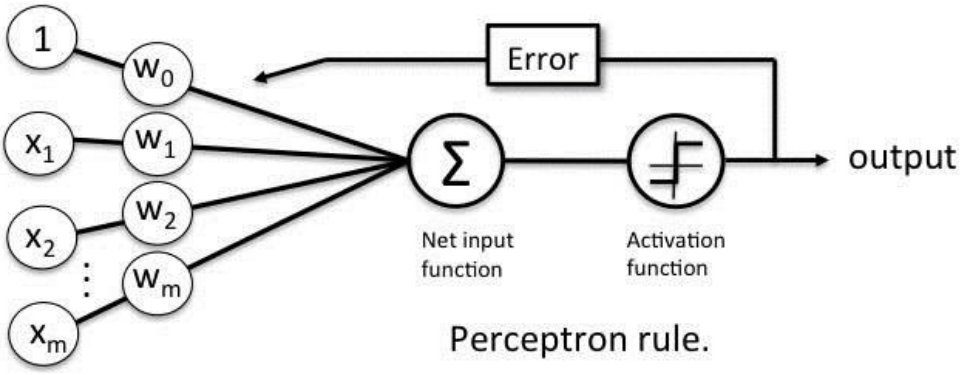
- "m" = number of inputs to the Perceptron

The output can be represented as "1" or "0." It can also be represented as "1" or "-1" depending on which activation function is used.

Let us learn the inputs of a perceptron in the next section.

Inputs of a Perceptron

A Perceptron accepts inputs, moderates them with certain weight values, then applies the transformation function to output the final result. The image below shows a Perceptron with a Boolean output.




A Boolean output is based on inputs such as salaried, married, age, past credit profile, etc. It has only two values:

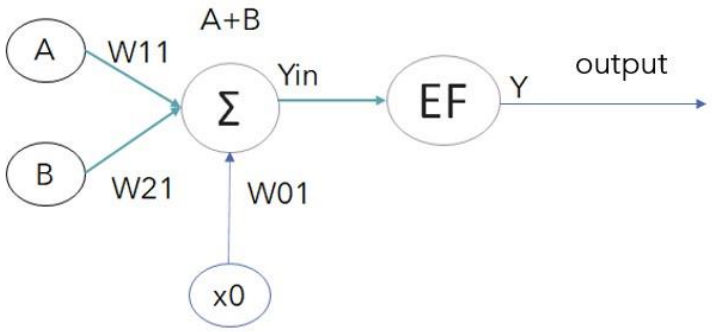
Yes and No or True and False. The summation function "∑" multiplies all inputs of "x" by weights "w" and then adds them up as follows:

$$w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

After calculating the output of summation function if is given as input to activation function ,here we use the activation function given as

$$y = \begin{cases} 1 & yin>0 \\ 0 & yin=0 \\ -1 & yin<0 \end{cases}$$

Evaluation function




(A+B) evaluation using 2 input perceptron

Algorithm:

1) Begin with random weights, then iteratively apply the perceptron to each training example, modifying the perceptron weights whenever it misclassifies an example.
2) This process is repeated, iterating through the training examples as many times as needed until the perceptron classifies all training examples correctly.
3) Weights are modified at each step according to the perceptron training rule, which revises the weight wi associated with input xi according to the rule.

$$w_i \leftarrow w_i + \Delta w_i$$

4) The role of the learning rate is to moderate the degree to which weights are changed at each step. It is usually set to some small value (e.g., 0.1) and is sometimes made to decay as the number of weight-tuning iterations increases

Where,

$$\Delta w_i = \eta(t-o)x_i$$

Here,
***t*** is the target output for the current training example
***o*** is the output generated by the perceptron
η is a positive constant called the ***learning rate***

## Results and Discussions

```
Python 3.7.3 Shell
File  Edit  Shell  Debug  Options  Window  Help
--------------------------------------------------------------------------------
---------------------------------
Evaluation of the Expression(A+B')
---------------------------------
Need to change the weights here.Now the updated Weights are  w0= 1 w1= -1 w2= -1
Need to change the weights here.Now the updated Weights are  w0= -1 w1= 1 w2= -3
NO need to change the Weights as Target = Output
Need to change the weights here.Now the updated Weights are  w0= 1 w1= 3 w2= -1
Need to change the weights here.Now the updated Weights are  w0= 3 w1= 1 w2= -3
NO need to change the Weights as Target = Output
NO need to change the Weights as Target = Output
NO need to change the Weights as Target = Output
--------------------------------------------------------------------------------
Weights after evaluation of (A+B') perceptron are w0= 3 w1= 1 w2= -3
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
---------------------------------
Evaluation of the Expression (A'.B)
---------------------------------
Need to change the weights here.Now the updated Weights are  w0= -1 w1= 1 w2= 1
Need to change the weights here.Now the updated Weights are  w0= 1 w1= -1 w2= 3
NO need to change the Weights as Target = Output
Need to change the weights here.Now the updated Weights are  w0= -1 w1= -3 w2= 1
Need to change the weights here.Now the updated Weights are  w0= -3 w1= -1 w2= 3
NO need to change the Weights as Target = Output
NO need to change the Weights as Target = Output
NO need to change the Weights as Target = Output
--------------------------------------------------------------------------------
Weights after evaluation of (A'.B) perceptron are w0= -3 w1= -1 w2= 3
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Need to change the weights here.Now the updated Weights are  w0= -1 w1= -1 w2= 1
Need to change the weights here.Now the updated Weights are  w0= -3 w1= 1 w2= -1
NO need to change the Weights as Target = Output
NO need to change the Weights as Target = Output
NO need to change the Weights as Target = Output
NO need to change the Weights as Target = Output
NO need to change the Weights as Target = Output
NO need to change the Weights as Target = Output
--------------------------------------------------------------------------------
Weights of the Hidder layer applied to final linear perceptron are w0= -3 w1= 1 w2= -1
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
The actual and perceptron output is same -1 -1
The actual and perceptron output is same
The actual and perceptron output is same -1 -1
The actual and perceptron output is same -1 -1
>>>
```

The first two perceptions are three input perceptron as shown in figure and the last perceptron is a 2 input perceptron with the inputs from the outputs of the first two perceptions .Each summation function output is given as input to the evaluation function ,as the evaluation function gives us the required output .There are three perceptions ,two with three inputs and one with two inputs and the all the weights for the three perceptions has been calculated by using the different functions for different weights and verified by calculating the outputs of the ANN using the weights calculated and verified with the actual outputs of the expression .