Assignment – 1

Find-S Algorithm

In order to understand Find-S algorithm, you need to have a basic idea of the following concepts as well:

1. Concept Learning
2. General Hypothesis
3. Specific Hypothesis

1. Concept Learning

Let's try to understand concept learning with a real-life example. Most of human learning is based on past instances or experiences. For example, we are able to identify any type of vehicle based on a certain set of features like make, model, etc., that are defined over a large set of features.

These special features differentiate the set of cars, trucks, etc from the larger set of vehicles. These features that define the set of cars, trucks, etc are known as concepts.

Similar to this, machines can also learn from concepts to identify whether an object belongs to a specific category or not. Any algorithm that supports concept learning requires the following:

- Training Data
- Target Concept
- Actual Data Objects

2. General Hypothesis

Hypothesis, in general, is an explanation for something. The general hypothesis basically states the general relationship between the major variables. For example, a general hypothesis for ordering food would be *I want a burger.*

G = { '?', '?', '?',.....'?'}

3. Specific Hypothesis

The specific hypothesis fills in all the important details about the variables given in the general hypothesis. The more specific details into the example given above would be *I want a cheeseburger with a chicken pepperoni filling with a lot of lettuce.*

S = {'Φ','Φ','Φ'...'Φ'}

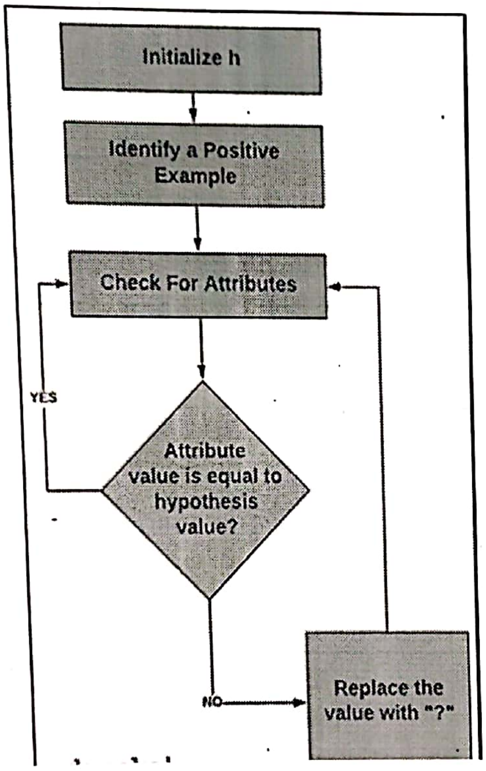
Now, let's talk about the Find-S Algorithm in Machine Learning.

The Find-S algorithm follows the steps written below:

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 '?'.

Now that we are done with the basic explanation of the Find-S algorithm, let us take a look at how it works.

How Does It Work?


Initialize h
Identify a Positive Example
Check For Attributes
YES
Attribute value is equal to hypothesis value?
NO
Replace the value with "?"

1. The process starts with initializing 'h' with the most specific hypothesis, generally, it is the first positive example in the data set.
2. We check for each positive example. If the example is negative, we will move on to the next example but if it is a positive example we will consider it for the next step.
3. We will check if each attribute in the example is equal to the hypothesis value.
4. If the value matches, then no changes are made.
5. If the value does not match, the value is changed to '?'.
6. We do this until we reach the last positive example in the data set.

Limitations of Find-S Algorithm

There are a few limitations of the Find-S algorithm listed down below:

1. There is no way to determine if the hypothesis is consistent throughout the data.
2. Inconsistent training sets can actually mislead the Find-S algorithm, since it ignores the negative examples.
3. Find-S algorithm does not provide a backtracking technique to determine the best possible changes that could be done to improve the resulting hypothesis.

Now that we are aware of the limitations of the Find-S algorithm, let us take a look at a practical implementation of the Find-S Algorithm.

Implementation of Find-S Algorithm

To understand the implementation, let us try to implement it to a smaller data set with a bunch of examples to decide if a person wants to go for a walk.

The concept of this particular problem will be on what days do a person likes to go on walk.

Time	Weather	Temperature	Company	Humidity	Wind	Goes
Morning	Sunny	Warm	Yes	Mild	Strong	Yes
Evening	Rainy	Cold	No	Mild	Normal	No
Morning	Sunny	Moderate	Yes	Normal	Normal	Yes
Evening	Sunny	Cold	Yes	High	Strong	Yes

Looking at the data set, we have six attributes and a final attribute that defines the positive or negative example. In this case, yes is a positive example, which means the person will go for a walk.

So now, the general hypothesis is:

h_0 = {'Morning', 'Sunny', 'Warm', 'Yes', 'Mild', 'Strong'}

This is our general hypothesis, and now we will consider each example one by one, but only the positive examples.

h_1= {'Morning', 'Sunny', '?', 'Yes', '?', '?'}

h_2 = {'?', 'Sunny', '?', 'Yes', '?', '?'}

We replaced all the different values in the general hypothesis to get a resultant hypothesis.

	Fever	Cough	Bt	Smell	Taste	Infect
0	NO	NO	NO	NO	No	0
1	Yes	Yes	Yes	Yes	Yes	1
2	Yes	Yes	No	Yes	NO	0
3	Yes	No	Yes	Yes	Yes	1
4	Yes	Yes	Yes	Yes	Yes	1
5	NO	Yes	NO	NO	Yes	0
6	Yes	No	Yes	Yes	Yes	1
7	Yes	NO	Yes	Yes	Yes	1
8	No	Yes	Yes	Yes	Yes	1
9	Yes	Yes	Yes	Yes	Yes	1
10	No	Yes	No	Yes	Yes	0
11	No	Yes	Yes	Yes	Yes	1
12	No	Yes	Yes	No	No	0
13	Yes	Yes	No	No	NO	0

Assignment-1

Objective: **Implement and Demonstrate Find-S Algorithm for finding the most specific hypothesis based on a given training data samples. Read the training data from .CSV file.**

Step-1: Importing necessory libraries.

- numpy for numerical operations.
- pandas for reading csv.

```
In [1]: import numpy as np
        import pandas as pd
```

Step-2: Reading csv file.

```
In [2]: df = pd.read_csv("./Covid.csv")
```

Step-3: Inspection of data and understanding it.

```
In [3]: df.head()
```

Out[3]:

	Fever	Cough	Breathing issues	Smell	taste	Infected
0	NO	NO	NO	NO	NO	0
1	YES	YES	YES	YES	YES	1
2	YES	YES	NO	YES	NO	0
3	YES	NO	YES	YES	YES	1
4	YES	YES	YES	YES	YES	1

About Code (Just for knowledge):

- This df.head() code prints first 5 lines of your dataframe.

About Data:

- Total 6 columns are there.
- Fever, Cough, Breathing issues, Smell and test are features.
- infected column is our target column.
- infected column has binary data. (1/0) representing +ve and -ve results.

```
In [4]: df.shape

Out[4]: (14, 6)
```

This shows that in our dataframe total 14 raws and 6 columns are available.

```
In [5]:  df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Fever             14 non-null     object
 1   Cough             14 non-null     object
 2   Breathing issues  14 non-null     object
 3   Smell             14 non-null     object
 4   taste             14 non-null     object
 5   Infected          14 non-null     int64
dtypes: int64(1), object(5)
memory usage: 800.0+ bytes
```

This shows details of your dataframe:

- Data type and non null value of each feature and target.
- memory usage of this dataframe.

Step-4: Seperating features and target

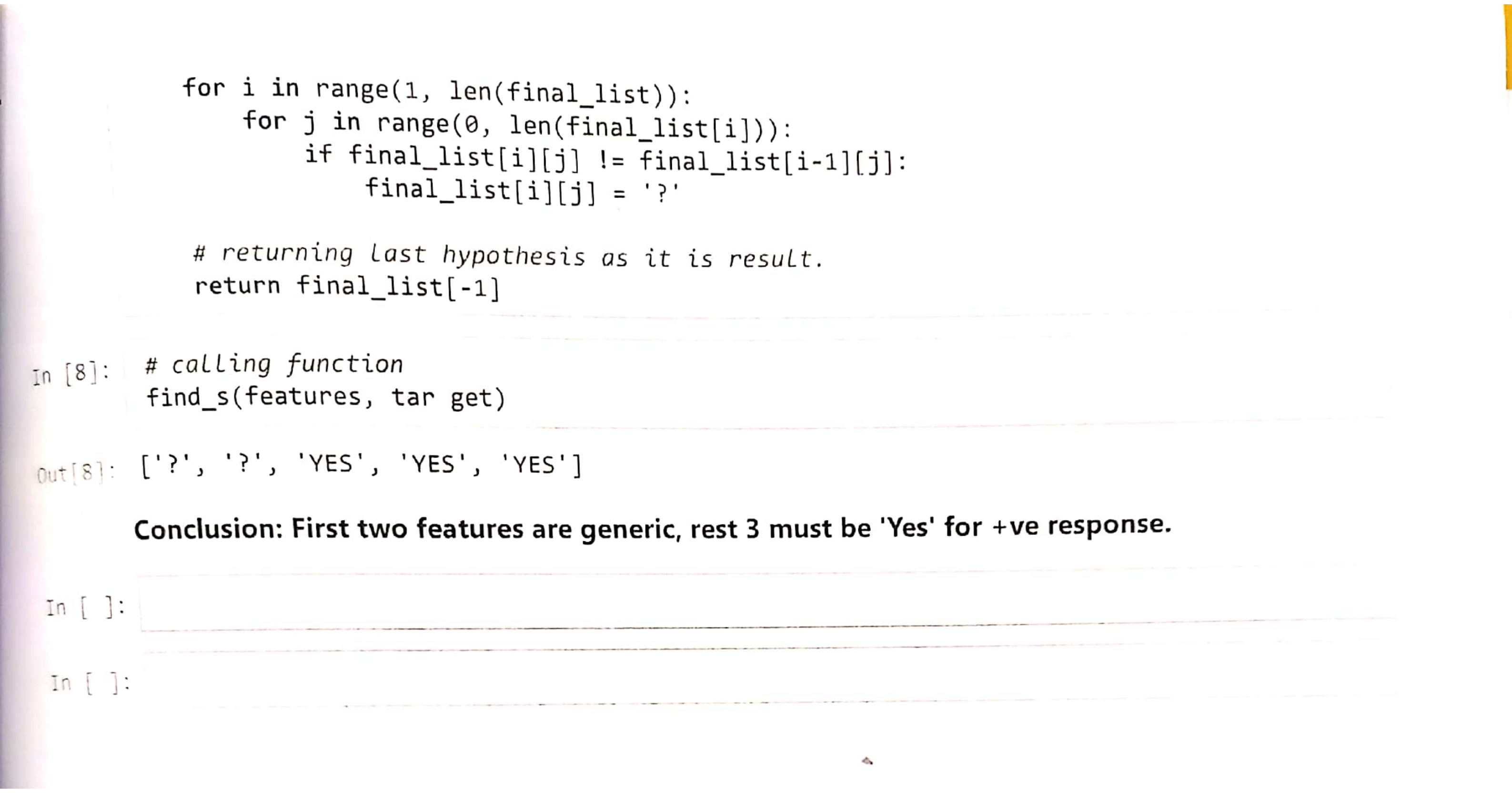
```
In [6]:  # features is all features where [:,:-1] represents [rows, columns] i.e. all rows and a
         features = np.array(df)[:,:-1]
         # target is last column where [:,-1] represents [rows, columns] i.e. all rows and last
         target = np.array(df)[:,-1]
```

Setp-5: Implementing Find-s algorithm

- finding first positive event.
- ho is equals to first positive event.
- find next positive event.
- for all features check, if this hypothesis matches with previous one, keep is same else replace it with '?' means generic.
- at the end last hypothesis h(n) will be your result.

```
In [7]:  def find_s(fet,tar):
             final_list = []
             # Running Loop to find +ve/-ve for all events/rows.
             for i, val_tar in enumerate(tar):
                 #Step-1: finding positive.
                 if val_tar == 1:
                     #Step-2: assigining in variable "current_hypothesis".
                     current_hypothesis = list(fet[i])
                     final_list.append(current_hypothesis)
                 else:
                     try:
                         final_list.append(final_list[i-1])
                     except:
                         final_list = final_list

             #Step-4: checking every other hypothesis to find if any previous value is same or n
```



```
    for i in range(1, len(final_list)):
        for j in range(0, len(final_list[i])):
            if final_list[i][j] != final_list[i-1][j]:
                final_list[i][j] = '?'

    # returning last hypothesis as it is result.
    return final_list[-1]
```

In [8]:

```
# calling function
find_s(features, tar get)
```

Out[8]:

```
['?', '?', 'YES', 'YES', 'YES']
```

Conclusion: First two features are generic, rest 3 must be 'Yes' for +ve response.

In [ ]:

In [ ]: