Sardar Patel Institute of Technology, Mumbai
Department of Electronics and Telecommunication Engineering
B.E. Sem-VII (2023-2024)
ETEL71A - Machine Learning and AI

Experiment: Find-S Algorithm

Name: Narayan Gawas **UID:** 2021201070

Objective: To explore how FIND-S algorithm is used for finding the most specific hypothesis based on a given set of training data samples.
Outcomes:

1. Representation of hypothesis
2. Apply Find-S algorithm on the given data to get the most specific hypothesis
3. Interpret the output of Find-S

System Requirements: Linux OS with Python and libraries or R or windows with MATLAB

Data Set Link: https://www.kaggle.com/datasets/imvickykumar999/find-s-algorithm-dataset

Algorithm:
Finds the most specific hypothesis matching the training
example (hence the name).

1. Initialize h to the most specific hypothesis in H
2. For each positive training instance x
   For each attribute constraint ai in h
   If the constraint ai in h is satisfied by x
   Then do nothing
   Else replace ai in h by the next more general constraint that is satisfied by x
3. Output hypothesis h

Code with Output:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/find-s-algorithm-dataset/ws.csv
```

```
data = pd.read_csv
('/kaggle/input/find-s-algorithm-dataset/ws.csv',names=['sky','Temp','Humi
dity','Wind','Water','Forecast','Enjoy Sport'])
Data
```

Out[2]:

	sky	Temp	Humidity	Wind	Water	Forecast	Enjoy Sport
0	Sunny	Warm	Normal	Strong	Warm	Same	Yes
1	Sunny	Warm	High	Strong	Warm	Same	Yes
2	Rainy	Cold	High	Strong	Warm	Change	No
3	Sunny	Warm	High	Strong	Cool	Change	Yes

```
h0 = np.zeros(6)
```

```
h0
```

Out[3]:

```
array([0., 0., 0., 0., 0., 0.])
```

```
data_column_dropped = data.drop(columns=['Enjoy Sport'],index=[2])
```

```
data_column_dropped
```

Out[4]:

	sky	Temp	Humidity	Wind	Water	Forecast
0	Sunny	Warm	Normal	Strong	Warm	Same
1	Sunny	Warm	High	Strong	Warm	Same
3	Sunny	Warm	High	Strong	Cool	Change

```
data_target = np.array(data)[:,-1]
```

```
data_target
```

Out[5]:

```
array(['Yes', 'Yes', 'No', 'Yes'], dtype=object)
```

```
data_attributes = np.array(data)[:,:-1]
```

```
data_attributes
```

Out[6]:

```
array([['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same'],
       ['Sunny', 'Warm', 'High', 'Strong', 'Warm', 'Same'],
       ['Rainy', 'Cold', 'High', 'Strong', 'Warm', 'Change'],
       ['Sunny', 'Warm', 'High', 'Strong', 'Cool', 'Change']],
      dtype=object)
```

data

Out[7]:

	sky	Temp	Humidity	Wind	Water	Forecast	Enjoy Sport
0	Sunny	Warm	Normal	Strong	Warm	Same	Yes
1	Sunny	Warm	High	Strong	Warm	Same	Yes
2	Rainy	Cold	High	Strong	Warm	Change	No
3	Sunny	Warm	High	Strong	Cool	Change	Yes

```
def train(c,t):
    #c - attributes, t = target
    for i, val in enumerate(t):
        if val == 'Yes':
            specific_hypothesis = c[i].copy()
            break
    for i,val in enumerate(c):
        if t[i] == 'Yes':
            for x in range (len(specific_hypothesis)):
                if val[x] != specific_hypothesis[x]:
                    specific_hypothesis[x] = '?'
                else:
                    pass
    return specific_hypothesis
print(f'The final hypothesis is - {train(data_attributes,data_target)}')
```

```
The final hypothesis is - ['Sunny' 'Warm' '?' 'Strong' '?' '?']
```

Interpretation of output: the Find-S algorithm is a simple machine learning technique that aims to find the most specific pattern or hypothesis from a set of positive examples. It generalizes patterns from data, handles incomplete information, and is used for educational purposes or in straightforward binary classification tasks. However, it has limitations in terms of expressiveness and may not be suitable for complex real-world problems.

Application:

Q1-What are the Limitations of Find-S Algorithms?
Ans:
Limited Expressiveness: Find-S can only represent hypotheses as conjunctions of attribute conditions and cannot handle complex relationships or disjunctions.
Lack of Negative Examples: It doesn't consider negative examples, making it unsuitable for problems where negative examples are crucial.
No Probabilistic Information: Find-S doesn't provide probabilistic information about the learned concept.
Sensitive to Order: The order in which examples are presented can affect the final hypothesis.
Assumes Consistency: It assumes that the concept being learned is consistent and does not change over time.
May Not Generalize Well: In some cases, the final hypothesis may be overly specific and may not generalize effectively to unseen data.

Q2-How many concepts are possible for this instance space of a given dataset?
Ans: The number of possible concepts in an instance space depends on the complexity and cardinality of the attributes in the dataset. Each combination of attribute values constitutes a potential concept.
Binary Attributes: If you have binary attributes (attributes that can take on only two values, e.g., True/False or 0/1), for each attribute, there are two possible concepts (presence or absence of the attribute). If you have 'n' binary attributes, you have 2^n possible concepts.
Categorical Attributes: For categorical attributes, the number of possible concepts depends on the number of unique values that attribute can take. If you have 'k' possible values for an attribute, there are k possible concepts for that attribute. Multiply this by the number of categorical attributes.
Continuous Attributes: For continuous attributes, the number of possible concepts is infinite, as there are infinitely many possible thresholds and ranges that can define a concept.
Mixed Attributes: If your dataset contains a mix of binary, categorical, and continuous attributes, you need to consider the possibilities for each attribute type and then multiply them together to get the total number of possible concepts.

Q3-How many hypotheses can be expressed by the hypothesis language?
Ans: the hypothesis language in Find-S allows you to express all possible combinations of literals for the attributes in your dataset using conjunctions.
Now, let's assume you have 'n' binary attributes, and for each attribute, you can have two possible literals (true or false). Therefore, for each attribute, you have 2 possible literals. In total, there are 2^n possible combinations of literals for 'n' attributes. Each of these combinations can be expressed as a hypothesis.
So, the number of hypotheses that can be expressed by the hypothesis language in the Find-S algorithm is 2^n, where 'n' is the number of binary attributes in your dataset.

Conclusion :
The Find-S algorithm is a straightforward and basic machine learning approach for binary classification tasks. It aims to find the most specific hypothesis that fits positive examples while ignoring negative ones. While easy to understand, it has limitations, like being sensitive to noise and limited to binary attributes. It's a foundational concept in machine learning but may not be suitable for complex real-world problems.