Abhishek Murthy
21BDS0064

Fall Sem 2024-2025

DA - 1

Machine Learning Lab

07-08-2024

Find-S Algorithm:

Explanation:

- ? (Question Mark):
  This symbol denotes a wildcard or placeholder, signifying that any value is acceptable for the corresponding attribute.

- Specific Value:
  Specific attribute values are explicitly defined within a hypothesis.

- φ (Phi):
  The symbol φ represents a null or empty value, indicating that no value is acceptable for the corresponding attribute.

- Most General Hypothesis:
  Denoted by {?, ?, ?, ?, ?}, the most general hypothesis encompasses all possible attribute values, offering a broad yet inclusive representation of the concept being learned.

- Most Specific Hypothesis:
  In stark contrast, the most specific hypothesis is represented by {φ, φ, φ, φ, φ}, where each attribute is constrained to null values.

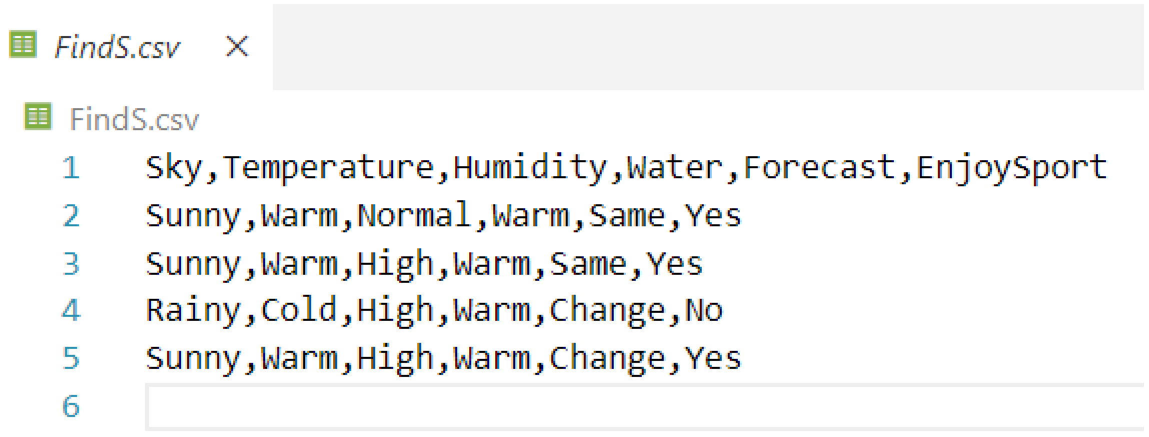
Steps:

1. Initialization: Start with the most specific hypothesis, S: <Ø, Ø, Ø, Ø, Ø>
2. Iteration through Training Examples: For each negative example, do nothing. For each positive example, generalize the hypothesis if it's too specific. Replace null (φ) values with attribute values from the example or use wildcard (?) symbols where needed.
   h1: <Sunny, Warm, Normal, Warm, Same>
   h2: <Sunny, Warm, ?, Warm, Same>
   h3: No updating since it's a negative example
   h4: <Sunny, Warm, ?, Warm, ?>
3. Convergence: Process all training examples, arriving at a final hypothesis that represents the learned concept based on the data.
4. Final Hypothesis: The algorithm's output is a hypothesis that encapsulates the patterns and regularities in the training data.
   h_final: <Sunny, Warm, ?, Warm, ?>

Code:

```
import pandas as pd

df = pd.read_csv('FindS.csv')

def find_s_algorithm(df):
    # set all initial hypotheses to 0
    hypothesis = ['0'] * (6)

    for i in range(len(df)):
        # consider only the positive instance
        if df.iloc[i, -1] == 'Yes':
            # compare the hypotheses with the actual values, if its equal then its satisfied so nothing to do
            # if its not equal then we need to change the hypothesis to the more general constraint
            for j in range(len(hypothesis)):
                actual_term = df.iloc[i, j]
                if hypothesis[j] == '0':
                    hypothesis[j] = actual_term
                elif hypothesis[j] != actual_term:
                    hypothesis[j] = '?'
    return hypothesis[:-1]

print(find_s_algorithm(df))
```

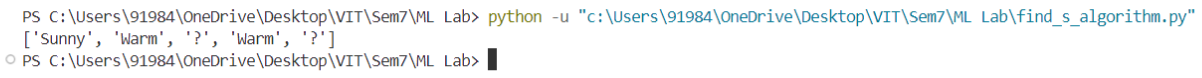
FindS.csv


FindS.csv ×

FindS.csv

```
Sky,Temperature,Humidity,Water,Forecast,EnjoySport
Sunny,Warm,Normal,Warm,Same,Yes
Sunny,Warm,High,Warm,Same,Yes
Rainy,Cold,High,Warm,Change,No
Sunny,Warm,High,Warm,Change,Yes
```

Output:



```
PS C:\Users\91984\OneDrive\Desktop\VIT\Sem7\ML Lab> python -u "c:\Users\91984\OneDrive\Desktop\VIT\Sem7\ML Lab\find_s_algorithm.py"
['Sunny', 'Warm', '?', 'Warm', '?']
PS C:\Users\91984\OneDrive\Desktop\VIT\Sem7\ML Lab>
```

Candidate Elimination Algorithm:

Explanation:

- ? (Question Mark):
  This symbol is a wildcard that can match any value for the corresponding attribute.

- Specific Value:
  These are explicit values for attributes in a hypothesis.

- φ (Phi):
  This symbol represents a null or empty value, indicating that no value is acceptable for the corresponding attribute.

- Most General Hypothesis:
  Represented as {?, ?, ?, ?, ?, ?}, this hypothesis includes all possible attribute values, being very inclusive.

- Most Specific Hypothesis:
  Represented as {φ, φ, φ, φ, φ, φ}, this hypothesis excludes all attribute values, being very restrictive.

Steps:

1. Initialization: Start with the most specific hypothesis $S_0 = \{\phi,\phi,\phi,\phi,\phi,\phi\}$ and the most general hypothesis $G_0 = \{?,?,?,?,?,?\}$
2. Iteration through Training Examples:

For each positive example: Check if the example is covered by the current hypothesis in S. If not, generalize S minimally so that it covers the example. Remove from G any hypothesis that does not cover the positive example.

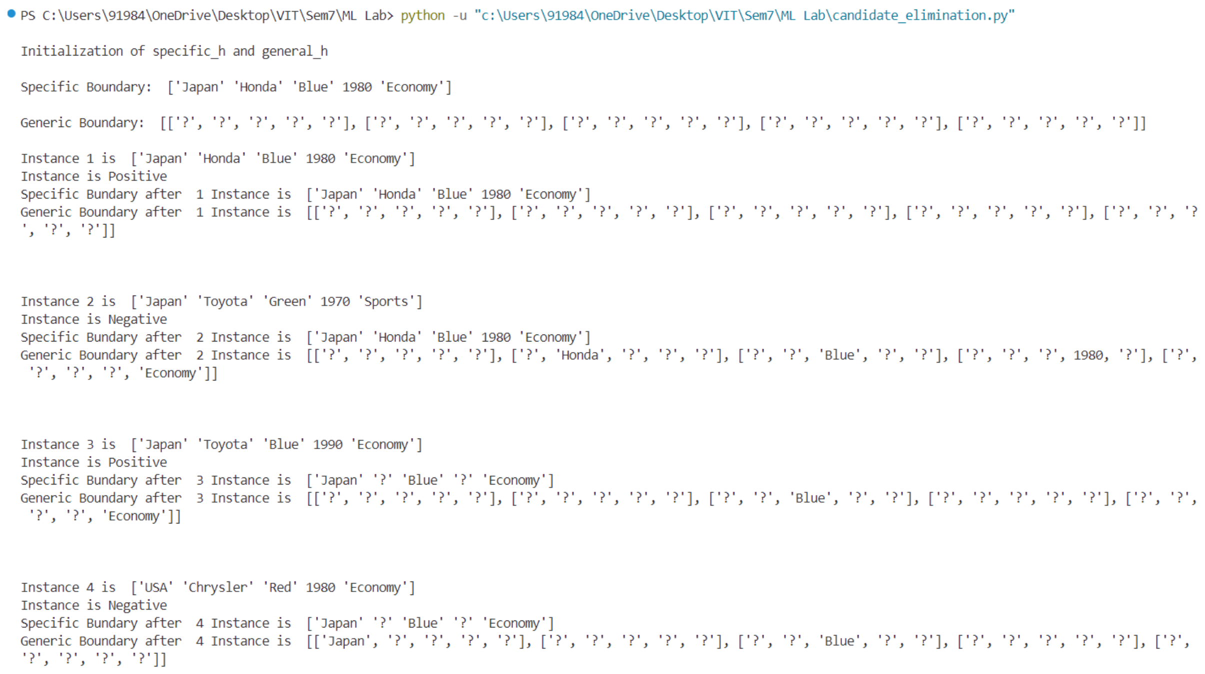
For each negative example: Remove from S any hypothesis that covers the negative example. Specialize hypotheses in G minimally so that they do not cover the negative example, ensuring they still cover all positive examples seen so far.

3. Iterative Refinement: Continue the process of refining S and G with each training example, ensuring that S becomes general enough to cover all positive examples and G becomes specific enough to exclude all negative examples.
4. Convergence: Once all training examples have been processed, S and G will converge to the boundaries of the version space, where S is the most specific hypothesis and G is the most general hypothesis consistent with the data.
5. Final Hypotheses: The final output is a set of hypotheses forming the version space, representing all possible hypotheses consistent with the observed data.

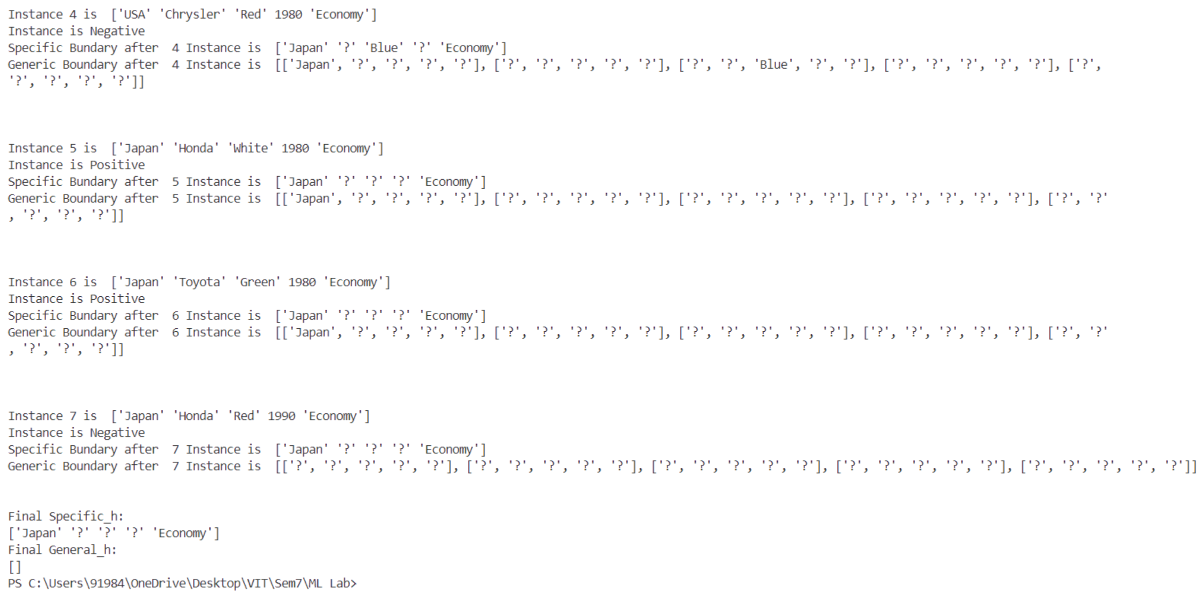
Code:

```
import numpy as np
import pandas as pd

data = pd.read_csv('CandidateElimination.csv')
# separate into concepts and target, i.e. features and target
concepts = np.array(data.iloc[:,0:-1])
target = np.array(data.iloc[:,-1])

def learn(concepts, target):

    # initialize the specific hypothesis to the first positive instance
    specific_h = concepts[0].copy()
    print("\nSpecific Boundary: ", specific_h)

    # initialize the general hypothesis to the most general instance
    general_h = [["?" for i in range(len(specific_h))] for i in range(len(specific_h))]
    print("\nGeneric Boundary: ",general_h)

    for i, h in enumerate(concepts):
        print("\nInstance", i+1 , "is ", h)
        if target[i] == "Positive":
            for x in range(len(specific_h)):
                # generalise specific_h to allow positive instance
                if h[x]!= specific_h[x]:
                    specific_h[x] ='?'
                    general_h[x][x] ='?'

        if target[i] == "Negative":
            for x in range(len(specific_h)):
                # specialise general_h to allow negative instance
                if h[x]!= specific_h[x]:
                    general_h[x][x] = specific_h[x]
                else:
                    general_h[x][x] = '?'
        # print out the boundaries
        print("Specific Boundary after ", specific_h)
        print("Generic Boundary after ", general_h)
        print("\n")

    indices = [i for i, val in enumerate(general_h) if val == ['?', '?', '?', '?', '?']]
    for i in indices:
        general_h.remove(['?', '?', '?', '?', '?'])
    return specific_h, general_h

s_final, g_final = learn(concepts, target)

print("Final Specific_h: ", s_final, sep="\n")
print("Final General_h: ", g_final, sep="\n")
```

Output:



```
● PS C:\Users\91984\OneDrive\Desktop\VIT\Sem7\ML Lab> python -u "c:\Users\91984\OneDrive\Desktop\VIT\Sem7\ML Lab\candidate_elimination.py"

 Initialization of specific_h and general_h

 Specific Boundary:  ['Japan' 'Honda' 'Blue' 1980 'Economy']

 Generic Boundary:  [['?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?']]

 Instance 1 is  ['Japan' 'Honda' 'Blue' 1980 'Economy']
 Instance is Positive
 Specific Bundary after  1 Instance is  ['Japan' 'Honda' 'Blue' 1980 'Economy']
 Generic Boundary after  1 Instance is  [['?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', '?
 ', '?', '?']]



 Instance 2 is  ['Japan' 'Toyota' 'Green' 1970 'Sports']
 Instance is Negative
 Specific Bundary after  2 Instance is  ['Japan' 'Honda' 'Blue' 1980 'Economy']
 Generic Boundary after  2 Instance is  [['?', '?', '?', '?', '?'], ['?', 'Honda', '?', '?', '?'], ['?', '?', 'Blue', '?', '?'], ['?', '?', '?', 1980, '?'], ['?',
  '?', '?', '?', 'Economy']]



 Instance 3 is  ['Japan' 'Toyota' 'Blue' 1990 'Economy']
 Instance is Positive
 Specific Bundary after  3 Instance is  ['Japan' '?' 'Blue' '?' 'Economy']
 Generic Boundary after  3 Instance is  [['?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', 'Blue', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?',
  '?', '?', 'Economy']]



 Instance 4 is  ['USA' 'Chrysler' 'Red' 1980 'Economy']
 Instance is Negative
 Specific Bundary after  4 Instance is  ['Japan' '?' 'Blue' '?' 'Economy']
 Generic Boundary after  4 Instance is  [['Japan', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', 'Blue', '?', '?'], ['?', '?', '?', '?', '?'], ['?',
 '?', '?', '?', '?']]
```



```
Instance 4 is  ['USA' 'Chrysler' 'Red' 1980 'Economy']
Instance is Negative
Specific Bundary after  4 Instance is  ['Japan' '?' 'Blue' '?' 'Economy']
Generic Boundary after  4 Instance is  [['Japan', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', 'Blue', '?', '?'], ['?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?']]



Instance 5 is  ['Japan' 'Honda' 'White' 1980 'Economy']
Instance is Positive
Specific Bundary after  5 Instance is  ['Japan' '?' '?' '?' 'Economy']
Generic Boundary after  5 Instance is  [['Japan', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?'
, '?', '?', '?']]



Instance 6 is  ['Japan' 'Toyota' 'Green' 1980 'Economy']
Instance is Positive
Specific Bundary after  6 Instance is  ['Japan' '?' '?' '?' 'Economy']
Generic Boundary after  6 Instance is  [['Japan', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?'
, '?', '?', '?']]



Instance 7 is  ['Japan' 'Honda' 'Red' 1990 'Economy']
Instance is Negative
Specific Bundary after  7 Instance is  ['Japan' '?' '?' '?' 'Economy']
Generic Boundary after  7 Instance is  [['?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?']]

Final Specific_h:
['Japan' '?' '?' '?' 'Economy']
Final General_h:
[]
PS C:\Users\91984\OneDrive\Desktop\VIT\Sem7\ML Lab>
```