Experiment 2 - Candidate-Elimination Algorithm

April 30, 2023

1 Experiment Details

1.1 Submitted By

Desh Iyer, 500081889, Year III, AI/ML(H), B5

1.2 Problem Statement

For a given set of training data examples stored in a `.csv` file, implement and demonstrate the candidate elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.

1.3 Theory

The Candidate-Elimination algorithm is a concept learning algorithm used to find the set of all hypotheses that are consistent with a given set of training data samples. The algorithm maintains two sets of hypotheses: the set of most specific hypotheses S and the set of most general hypotheses G. Initially, S contains the most specific hypothesis possible, and G contains the most general hypothesis possible. The algorithm iteratively updates S and G based on the training data.

Here are the terms used in the description of the algorithm.

- A hypothesis h is a conjunction of n literals, where each literal can take one of two values: true or false.
- A training example is a pair (x, y), where x is an n-dimensional vector of attribute values, and y is either true or false.
- A positive example is a training example where y is true.
- A negative example is a training example where y is false.

1.4 Advantages

- The algorithm can handle noisy data and errors in the training data.
- The algorithm outputs a set of hypotheses that are consistent with the training data, which can be used for further analysis or decision making.

1.5 Limitations

- The algorithm can become computationally expensive when the number of attributes or the size of the hypothesis space is large.
- The algorithm does not provide any measure of the quality or confidence of the hypotheses.
- The algorithm assumes that the target concept is represented by a single consistent hypothesis.

1.6 Pseudocode

The Candidate-Elimination algorithm starts by initializing S with the most specific hypothesis possible, and G with the most general hypothesis possible:

```
S ← { < , , ..., > }
G ← { < ?, ?, ..., ? > }
```

Then, for each training example (x, y) in the training data, the algorithm updates S and G based on whether the example is positive or negative. For positive examples, the algorithm updates S to include only the hypotheses that are consistent with the example, and updates G to remove any hypotheses that are inconsistent with the example. For negative examples, the algorithm updates G to include only the hypotheses that are consistent with the example, and updates S to remove any hypotheses that are inconsistent with the example.

```
for each training example (x, y) do
    if y = true then
        S ← { h belongs to S : h(x) = y } # Keep only the hypotheses that are consistent with x
        for g belongs to G do
        if g(x) != y and g is still in G then
            G ← G - { g } # Remove any hypotheses that are inconsistent with x.
            # add to G all minimal generalizations of h
            # that are consistent with all positive training examples seen so far
        end if
        end for
    else # y = false
        G ← { h  G : h(x) != y } # Keep only the hypotheses that are consistent with x.
        for s belongs to S do
        if s(x) = y and s is still in S then
            S ← S - { s } # Remove any hypotheses that are inconsistent with x.
            # add to S all minimal specializations of h
            # that are consistent with all negative training examples seen so far
        end if
        end for
    end if
end for
```

2 Import Libraries

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

3 Extract Data

[ ]:
```
data = pd.read_csv('../data/candidate-elimination.csv')

concepts = np.array(data.iloc[:, 0:-1])
target = np.array(data.iloc[:, -1])
```

```
print("\nInstances are:\n",concepts)
print("\nTarget values are:\n",target)
```

```
Instances are:
 [['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']
 ['Rainy' 'Cold' 'High' 'Strong' 'Warm' 'Change']
 ['Sunny' 'Warm' 'High' 'Strong' 'Cool' 'Change']]

Target values are:
 ['Yes' 'No' 'Yes']
```

4 Define Function to Learn Dataset

[ ]:
```
def learn(concepts, target):
    specific_h = concepts[0]
    general_h = [["?" for i in range(len(specific_h))] for i in␣
 ↪range(len(specific_h))]

    print("\nInitializing hypotheses")
    print("\nSpecific Boundary: ", specific_h)
    print("\nGeneric Boundary:\n",general_h)

    for i, h in enumerate(concepts):
        print("\nInstance", i + 1 , "is ", h)
        if target[i] == "yes":
            print("Instance is Positive")
            for x in range(len(specific_h)):
                if h[x] != specific_h[x]:
                    specific_h[x] ='?'
                    general_h[x][x] ='?'

        if target[i] == "no":
            print("Instance is Negative ")
            for x in range(len(specific_h)):
                if h[x] != specific_h[x]:
                    general_h[x][x] = specific_h[x]
                else:
                    general_h[x][x] = '?'

        print("Specific boundary after", i + 1, "iteration is ", specific_h)
        print("Generic boundary after", i + 1, "iteration is ", general_h)

    indices = [i for i, val in enumerate(general_h) if val == ['?', '?', '?', '?
 ↪', '?', '?']]
```

```
    for i in indices:
        general_h.remove(['?', '?', '?', '?', '?', '?'])
    return specific_h, general_h
```

5 Generate Hypotheses

[ ]:
```
s_final, g_final = learn(concepts, target)

print("\nFinal Specific Hypothesis: ", s_final, sep="\n")
print("Final General Hypothesis: ", g_final, sep="\n")
```

```
Initializing hypotheses

Specific Boundary:  ['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']

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

Instance 1 is  ['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']
Specific boundary after 1 iteration is  ['Sunny' 'Warm' 'High' 'Strong' 'Warm'
'Same']
Generic boundary after 1 iteration is  [['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Instance 2 is  ['Rainy' 'Cold' 'High' 'Strong' 'Warm' 'Change']
Specific boundary after 2 iteration is  ['Sunny' 'Warm' 'High' 'Strong' 'Warm'
'Same']
Generic boundary after 2 iteration is  [['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Instance 3 is  ['Sunny' 'Warm' 'High' 'Strong' 'Cool' 'Change']
Specific boundary after 3 iteration is  ['Sunny' 'Warm' 'High' 'Strong' 'Warm'
'Same']
Generic boundary after 3 iteration is  [['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Final Specific Hypothesis:
['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']
Final General Hypothesis:
[]
```