FindS_and_Candidate_Elimination_Algorithm_p1

November 1, 2025

1. Read a dataset from the user and i. Use the Find-S algorithm to find the most specific hypothesis that is consistent with the positive examples. Ii. What is the final hypothesis after processing all the positive examples? Using the same dataset, apply the Candidate Elimination algorithm. Determine the final version space after processing all examples (both positive and negative). What are the most specific and most general hypotheses in the version space?

[3]:
```python
import pandas as pd

# ---------- Step 1: Read Dataset ----------
print("Enter dataset CSV file path (or press Enter to use default sample): ")
file_path = input().strip()

if file_path:
    data = pd.read_csv(file_path)
else:
    # Default dataset (EnjoySport)
    data = pd.DataFrame([
        ['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same', 'Yes'],
        ['Sunny', 'Warm', 'High', 'Strong', 'Warm', 'Same', 'Yes'],
        ['Rainy', 'Cold', 'High', 'Strong', 'Warm', 'Change', 'No'],
        ['Sunny', 'Warm', 'High', 'Strong', 'Cool', 'Change', 'Yes']
    ], columns=['Sky', 'AirTemp', 'Humidity', 'Wind', 'Water', 'Forecast',␣
 ↪'EnjoySport'])

print("\nDataset:\n", data)

attributes = data.columns[:-1]
target = data.columns[-1]

# ---------- Step 2: FIND-S Algorithm ----------
def find_s_algorithm(data):
    specific_h = ['0'] * len(attributes)

    for i in range(len(data)):
        if data[target][i].lower() == 'yes':
            if specific_h[0] == '0':  # first positive example
                specific_h = data.iloc[i, :-1].tolist()
            else:
                for j in range(len(specific_h)):
                    if specific_h[j] != data.iloc[i, j]:
                        specific_h[j] = '?'
    return specific_h

specific_hypothesis = find_s_algorithm(data)
print("\n=== FIND-S Algorithm ===")
print("Most specific hypothesis:", specific_hypothesis)

# ---------- Step 3: Candidate Elimination Helpers ----------
def more_general(h1, h2):
    return all(x == '?' or x == y for x, y in zip(h1, h2))

def generalize_S(example, S):
    for h in S:
        for i in range(len(h)):
            if h[i] != example[i]:
                h[i] = '?'
    return S

def specialize_G(example, G, domains):
    new_G = []
    for h in G:
        for i in range(len(h)):
            if h[i] == '?':
                for value in domains[i]:
                    if value != example[i]:
                        new_h = h.copy()
                        new_h[i] = value
                        new_G.append(new_h)
    return new_G

# ---------- Step 4: Candidate Elimination Algorithm ----------
def candidate_elimination(data):
    domains = [list(data[attr].unique()) for attr in attributes]
    S = [['0'] * len(attributes)]
    G = [['?'] * len(attributes)]

    for i, row in data.iterrows():
        example = row[:-1].tolist()
        label = row[-1].lower()

        if label == 'yes':
            # Remove inconsistent hypotheses from G
            G = [g for g in G if more_general(g, example)]
            # Generalize S
            if S[0] == ['0'] * len(attributes):
                S[0] = example
            else:
                S = generalize_S(example, S)
        else:
            # Specialize G
            G = specialize_G(example, G, domains)
            # Remove inconsistent hypotheses from G
            G = [g for g in G if any(more_general(g, s) for s in S)]

    return S, G

S_final, G_final = candidate_elimination(data)

# ---------- Step 5: Output Results ----------
print("\n=== Candidate Elimination Algorithm ===")
print("Final Specific Boundary (S):", S_final)
print("Final General Boundary (G):", G_final)

print("\n=== Summary ===")
print("Most Specific Hypothesis:", S_final)
print("Most General Hypothesis:", G_final)
print("\nVersion Space contains all hypotheses between S and G that are␣
 ↪consistent with the training examples.")
```

```
Enter dataset CSV file path (or press Enter to use default sample):



Dataset:
     Sky AirTemp Humidity   Wind Water Forecast EnjoySport
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

=== FIND-S Algorithm ===
Most specific hypothesis: ['Sunny', 'Warm', '?', 'Strong', '?', '?']

=== Candidate Elimination Algorithm ===
Final Specific Boundary (S): [['Sunny', 'Warm', '?', 'Strong', '?', '?']]
Final General Boundary (G): [['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm',
'?', '?', '?', '?']]

=== Summary ===
Most Specific Hypothesis: [['Sunny', 'Warm', '?', 'Strong', '?', '?']]
Most General Hypothesis: [['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?',
'?', '?', '?']]
```

Version Space contains all hypotheses between S and G that are consistent with the training examples.

[ ]:

```
```