# Assignment-5

**Name:**Pothu Rahul

**Batch:**12

**Hall ticket no:**2203A51439

1. Implement a Hidden Markov Model (HMM) for weather forecasting, taking scenario where we have hidden weather states (e.g., "Sunny" and "Rainy") and observable events (e.g., "Dry", "Damp", "Wet"). The goal is to predict the sequence of weather states based on the observations.

```
!pip install hmmlearn
```

```
Requirement already satisfied: hmmlearn in /usr/local/lib/python3.10/dist-packages (0.3.2)
Requirement already satisfied: numpy>=1.10 in /usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.26.4)
Requirement already satisfied: scikit-learn!=0.22.0,>=0.16 in /usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.3.2)
Requirement already satisfied: scipy>=0.19 in /usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.13.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn!=0.22.0,>=0.16->hmmlearn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn!=0.22.0,>=0.16->hmmlearn) (3.5.0)
```

```python
import numpy as np

states = ["Sunny", "Rainy"]
observations = ["Dry", "Damp", "Wet"]

state_map = {state: i for i, state in enumerate(states)}
obs_map = {obs: i for i, obs in enumerate(observations)}

transition_prob = np.array([
    [0.7, 0.3],
    [0.4, 0.6]
])

emission_prob = np.array([
    [0.6, 0.3, 0.1],
    [0.2, 0.4, 0.4]
])

start_prob = np.array([0.8, 0.2])

def viterbi(obs_sequence, start_prob, trans_prob, emit_prob):
    n_states = len(start_prob)
    n_obs = len(obs_sequence)

    viterbi_matrix = np.zeros((n_states, n_obs))
    backpointer = np.zeros((n_states, n_obs), dtype=int)

    viterbi_matrix[:, 0] = start_prob * emit_prob[:, obs_map[obs_sequence[0]]]

    for t in range(1, n_obs):
        for s in range(n_states):
            prob = viterbi_matrix[:, t - 1] * trans_prob[:, s] * emit_prob[s, obs_map[obs_sequence[t]]]
            viterbi_matrix[s, t] = np.max(prob)
            backpointer[s, t] = np.argmax(prob)

    best_path = np.zeros(n_obs, dtype=int)
    best_path[-1] = np.argmax(viterbi_matrix[:, -1])

    for t in range(n_obs - 2, -1, -1):
        best_path[t] = backpointer[best_path[t + 1], t + 1]

    best_state_sequence = [states[state] for state in best_path]

    return best_state_sequence, viterbi_matrix

obs_sequence = ["Dry", "Damp", "Wet"]

best_state_sequence, viterbi_matrix = viterbi(obs_sequence, start_prob, transition_prob, emission_prob)

print("Observation Sequence:", obs_sequence)
print("Best State Sequence:", best_state_sequence)
print("Viterbi Matrix:\n", viterbi_matrix)
```

```
Observation Sequence: ['Dry', 'Damp', 'Wet']
Best State Sequence: ['Sunny', 'Rainy', 'Rainy']
Viterbi Matrix:
 [[0.48     0.1008   0.007056]
 [0.04     0.0576   0.013824]]
```

## 2. Problem Setup

**States (S):** Hidden states representing the weather (e.g., "Sunny", "Rainy").

**Observations (O):** Observable events that we can measure (e.g., "Dry", "Damp", "Wet").

```
import numpy as np
from hmmlearn import hmm

states = ["Sunny", "Rainy"]
n_states = len(states)

observations = ["Dry", "Damp", "Wet"]
n_observations = len(observations)
print("States:", states)
print("Number of States:", n_states)
print("Observations:", observations)
print("Number of Observations:", n_observations)
```

```
States: ['Sunny', 'Rainy']
Number of States: 2
Observations: ['Dry', 'Damp', 'Wet']
Number of Observations: 3
```

**Transition Probabilities (A):** Th e probability of transitioning from one w ather state to another.

```
import numpy as np
from hmmlearn import hmm
states = ["Sunny", "Rainy"]
n_states = len(states)
observations = ["Dry", "Damp", "Wet"]
n_observations = len(observations)
transition_prob = np.array([
    [0.7, 0.3],  # Sunny -> Sunny, Sunny -> Rainy
    [0.4, 0.6]   # Rainy -> Sunny, Rainy -> Rainy
])
print("Transition Probabilities (A):")
print(transition_prob)
```

```
Transition Probabilities (A):
[[0.7 0.3]
 [0.4 0.6]]
```

**Emission Probabilities (B):** The probability of observing a certain event given the weather state.

```
import numpy as np
from hmmlearn import hmm

states = ["Sunny", "Rainy"]
n_states = len(states)

observations = ["Dry", "Damp", "Wet"]
n_observations = len(observations)
emission_prob = np.array([
    [0.8, 0.15, 0.05],  # Sunny -> Dry, Sunny -> Damp, Sunny -> Wet
    [0.1, 0.35, 0.55]   # Rainy -> Dry, Rainy -> Damp, Rainy -> Wet
])
print("Emission Probabilities (B):")
print(emission_prob)
```

```
Emission Probabilities (B):
[[0.8  0.15 0.05]
 [0.1  0.35 0.55]]
```

**Initial Probabilities ($\pi$):** The probability distribution over the initial states.

```
import numpy as np
states = ["Sunny", "Rainy"]
n_states = len(states)
start_prob = np.array([0.6, 0.4])  # Initial probability of being Sunny or Rainy
print("Initial Probabilities (π):")
print(start_prob)
```

```
Initial Probabilities (π):
[0.6 0.4]
```

2. **Take** different transition and emission probabilities or observation sequences on above problem and see how the model's predictions change.

```python
import numpy as np
from hmmlearn import hmm

states = ["Sunny", "Rainy"]
n_states = len(states)

transition_prob = np.array([
    [0.6, 0.4],
    [0.5, 0.5]
])

means = np.array([
    [1.2],
    [2.8]
])
covariances = np.array([
    [0.4],
    [0.6]
])

start_prob = np.array([0.7, 0.3])

model = hmm.GaussianHMM(n_components=n_states, covariance_type='diag')

model.startprob_ = start_prob
model.transmat_ = transition_prob
model.means_ = means
model.covars_ = covariances

observation_sequences = [
    np.array([1, 2, 3, 1, 2]).reshape(-1, 1),
    np.array([3, 3, 2, 1]).reshape(-1, 1)
]

for i, observation_sequence in enumerate(observation_sequences):

    logprob, hidden_states = model.decode(observation_sequence, algorithm="viterbi")
    print(f"\nObservation Sequence {i + 1}: {[observations[j-1] for j in observation_sequence.flatten()]}")
    print("Predicted Hidden States:", [states[i] for i in hidden_states])
```

```
Observation Sequence 1: ['Dry', 'Damp', 'Wet', 'Dry', 'Damp']
Predicted Hidden States: ['Sunny', 'Sunny', 'Rainy', 'Sunny', 'Sunny']

Observation Sequence 2: ['Wet', 'Wet', 'Damp', 'Dry']
Predicted Hidden States: ['Rainy', 'Rainy', 'Sunny', 'Sunny']
```