200010021

October 22, 2022

1 LAB 10 : Hidden Markov Model

[1]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Please refer to the following article to understand Hidden Markov Model

Here we will be dealing with 3 major problems :

1. Evaluation Problem
2. Learning Problem
3. Decoding Problem

1. Evaluation Problem : Implementation of Forward and Backward Algorithm

[2]:
```python
data = pd.read_csv('data_python.csv') ## Read the data, change the path␣
↪accordingly

V = data['Visible'].values

# Transition Probabilities
a = np.array(((0.54, 0.46), (0.49, 0.51)))

# Emission Probabilities
b = np.array(((0.16, 0.26, 0.58), (0.25, 0.28, 0.47)))

# Equal Probabilities for the initial distribution
initial_distribution = np.array((0.5, 0.5))


def forward(V, a, b, initial_distribution):
    alpha = np.zeros((V.shape[0], a.shape[0]))

    ## Write your code here
    alpha[0, :] = initial_distribution * b[:, V[0]]

    for t in range(1, V.shape[0]):
        for j in range(a.shape[0]):
```

```python
            alpha[t, j] = alpha[t - 1].dot(a[:, j]) * b[j, V[t]]

    return alpha


alpha = forward(V, a, b, initial_distribution)


def backward(V, a, b):
    beta = np.zeros((V.shape[0], a.shape[0]))

    ## Write your code here
    beta[V.shape[0] - 1] = np.ones((a.shape[0]))

    # Loop in backward way from T-1 to
    # Due to python indexing the actual loop will be T-2 to 0
    for t in range(V.shape[0] - 2, -1, -1):
        for j in range(a.shape[0]):
            beta[t, j] = (beta[t + 1] * b[:, V[t + 1]]).dot(a[j, :])

    return beta


beta = backward(V, a, b)
```

2. Learning Problem : Implementation of Baum Welch Algorithm

[3]:
```python
def baum_welch(V, a, b, initial_distribution, n_iter=100):
    M = a.shape[0]
    T = len(V)

    for n in range(n_iter):
        alpha = forward(V, a, b, initial_distribution)
        beta = backward(V, a, b)

        xi = np.zeros((M, M, T - 1))
        for t in range(T - 1):
            denominator = np.dot(np.dot(alpha[t, :].T, a) * b[:, V[t + 1]].T,␣
↪beta[t + 1, :])
            for i in range(M):
                numerator = alpha[t, i] * a[i, :] * b[:, V[t + 1]].T * beta[t +␣
↪1, :].T
                xi[i, :, t] = numerator / denominator

        gamma = np.sum(xi, axis=1)
        a = np.sum(xi, 2) / np.sum(gamma, axis=1).reshape((-1, 1))
```

```
        # Add additional T'th element in gamma
        gamma = np.hstack((gamma, np.sum(xi[:, :, T - 2], axis=0).reshape((-1,␣
↪1))))

        K = b.shape[1]
        denominator = np.sum(gamma, axis=1)
        for l in range(K):
            b[:, l] = np.sum(gamma[:, V == l], axis=1)

        b = np.divide(b, denominator.reshape((-1, 1)))

    return (a,b)

data = pd.read_csv('data_python.csv')

V = data['Visible'].values

# Transition Probabilities
a = np.ones((2, 2))
a = a / np.sum(a, axis=1)

# Emission Probabilities
b = np.array(((1, 3, 5), (2, 4, 6)))
b = b / np.sum(b, axis=1).reshape((-1, 1))

# Equal Probabilities for the initial distribution
initial_distribution = np.array((0.5, 0.5))

a,b = baum_welch(V, a, b, initial_distribution, n_iter=100)
```

3. Decoding Problem : Implementation of Viterbi Algorithm

[4]:
```
def viterbi(V, a, b, initial_distribution):
    T = V.shape[0]
    M = a.shape[0]

    omega = np.zeros((T, M))
    omega[0, :] = np.log(initial_distribution * b[:, V[0]])

    prev = np.zeros((T - 1, M))

    for t in range(1, T):
        for j in range(M):
            # Same as Forward Probability
            probability = omega[t - 1] + np.log(a[:, j]) + np.log(b[j, V[t]])

            # This is our most probable state given previous state at time t (1)
```

```
            prev[t - 1, j] = np.argmax(probability)

            # This is the probability of the most probable state (2)
            omega[t, j] = np.max(probability)

    # Path Array
    S = np.zeros(T)

    # Find the most probable last hidden state
    last_state = np.argmax(omega[T - 1, :])

    S[0] = last_state

    backtrack_index = 1
    for i in range(T - 2, -1, -1):
        S[backtrack_index] = prev[i, int(last_state)]
        last_state = prev[i, int(last_state)]
        backtrack_index += 1

    # Flip the path array since we were backtracking
    S = np.flip(S, axis=0)

    # Convert numeric values to actual hidden states
    result = []
    for s in S:
        if s == 0:
            result.append("A")
        else:
            result.append("B")
  ## Write your code here

    return result


data = pd.read_csv('data_python.csv')

V = data['Visible'].values

# Transition Probabilities
a = np.ones((2, 2))
a = a / np.sum(a, axis=1)

# Emission Probabilities
b = np.array(((1, 3, 5), (2, 4, 6)))
b = b / np.sum(b, axis=1).reshape((-1, 1))

# Equal Probabilities for the initial distribution
```

```
initial_distribution = np.array((0.5, 0.5))

a, b = baum_welch(V, a, b, initial_distribution, n_iter=100)

result = viterbi(V, a, b, initial_distribution)
```

4. Use the built-in **hmmlearn** package to fit the data and generate the result using the decoder

[5]:
```
%pip install hmmlearn
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: hmmlearn in
/home/abhishekj/.local/lib/python3.9/site-packages (0.2.7)
Requirement already satisfied: scikit-learn>=0.16 in
/home/abhishekj/.local/lib/python3.9/site-packages (from hmmlearn) (1.1.2)
Requirement already satisfied: numpy>=1.10 in
/home/abhishekj/.local/lib/python3.9/site-packages (from hmmlearn) (1.23.2)
Requirement already satisfied: scipy>=0.19 in
/home/abhishekj/.local/lib/python3.9/site-packages (from hmmlearn) (1.9.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/home/abhishekj/.local/lib/python3.9/site-packages (from scikit-
learn>=0.16->hmmlearn) (3.1.0)
Requirement already satisfied: joblib>=1.0.0 in
/home/abhishekj/.local/lib/python3.9/site-packages (from scikit-
learn>=0.16->hmmlearn) (1.1.0)
WARNING: Value for scheme.platlib does not match. Please report this to

<https://github.com/pypa/pip/issues/10151>

distutils: /home/abhishekj/.local/lib/python3.9/site-packages

sysconfig: /home/abhishekj/.local/lib64/python3.9/site-packages
WARNING: Additional context:

user = True

home = None

root = None

prefix = None
WARNING: You are using pip version 21.2.3; however, version 22.3 is

available.

You should consider upgrading via the '/bin/python -m pip install --upgrade pip'

command.
Note: you may need to restart the kernel to use updated packages.
```

[6]:
```
## Write your code here
from hmmlearn import hmm
import numpy as np
```

```python
import math

data = pd.read_csv('data_python.csv')

V = data['Visible'].values
```

[7]:
```python
print(V.shape)
V_reshaped = np.array(V.reshape(-1,1)).T
print(V_reshaped.shape)
```

```
(500,)
(1, 500)
```

[8]:
```python
model = hmm.MultinomialHMM(n_components=2)

model.startprob_ = np.array([0.5, 0.5])
model.transmat_ = np.array([[0.5, 0.5],
                            [0.5, 0.5]])
model.emissionprob_ = np.array([[0.11111111, 0.33333333, 0.55555556],
                                [0.16666667, 0.33333333 ,0.5]])

logprob, sequence = model.decode(V_reshaped)
out = []
for i in sequence:
  if i == 1:
    i = "B"
  else:
    i = "A"
  out.append(i)
print(out)
```

```
['B', 'B', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A',
'B', 'A', 'A', 'A', 'A', 'B', 'B', 'B', 'B', 'A', 'B', 'A', 'B', 'A', 'B', 'B',
'A', 'B', 'A', 'B', 'A', 'B', 'A', 'A', 'B', 'A', 'A', 'A', 'B', 'B', 'B', 'B',
'B', 'A', 'A', 'A', 'A', 'B', 'A', 'A', 'A', 'B', 'A', 'B', 'B', 'B', 'B', 'A',
'B', 'A', 'B', 'B', 'B', 'B', 'A', 'B', 'B', 'B', 'B', 'A', 'B', 'B', 'A', 'B',
'B', 'B', 'A', 'B', 'B', 'B', 'A', 'B', 'B', 'B', 'A', 'B', 'A', 'B', 'B', 'A',
'B', 'A', 'A', 'A', 'B', 'A', 'B', 'A', 'B', 'B', 'B', 'A', 'A', 'B', 'A', 'A',
'B', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'B', 'B', 'B', 'B', 'B', 'B', 'A', 'B',
'B', 'B', 'B', 'B', 'B', 'B', 'A', 'B', 'A', 'A', 'B', 'B', 'B', 'B', 'B', 'A',
'A', 'B', 'A', 'B', 'B', 'B', 'A', 'A', 'A', 'A', 'A', 'B', 'A', 'A', 'A', 'A',
'A', 'B', 'A', 'B', 'A', 'B', 'A', 'B', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A',
'B', 'A', 'B', 'A', 'B', 'B', 'B', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'B', 'A',
'A', 'A', 'A', 'A', 'B', 'A', 'B', 'A', 'B', 'A', 'B', 'A', 'B', 'B', 'A', 'B',
'B', 'B', 'B', 'B', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'B', 'B', 'B',
'B', 'A', 'B', 'B', 'A', 'A', 'B', 'A', 'A', 'A', 'B', 'B', 'B', 'A', 'A', 'A',
'B', 'B', 'B', 'B', 'A', 'A', 'B', 'A', 'A', 'A', 'B', 'A', 'A', 'A', 'A', 'B',
'A', 'B', 'B', 'B', 'A', 'B', 'B', 'A', 'A', 'A', 'B', 'B', 'B', 'B', 'B', 'A',
```

```
'B', 'A', 'B', 'B', 'A', 'B', 'A', 'A', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B',
'A', 'A', 'B', 'B', 'A', 'A', 'A', 'B', 'B', 'B', 'A', 'A', 'B', 'B', 'A', 'A',
'A', 'A', 'A', 'A', 'B', 'B', 'A', 'A', 'B', 'A', 'B', 'A', 'A', 'A', 'B', 'A',
'A', 'A', 'B', 'B', 'B', 'B', 'A', 'B', 'B', 'B', 'A', 'A', 'B', 'B', 'A', 'B',
'B', 'A', 'B', 'B', 'B', 'B', 'B', 'A', 'B', 'A', 'B', 'A', 'A', 'A', 'B', 'A',
'B', 'B', 'A', 'A', 'A', 'A', 'A', 'A', 'B', 'B', 'B', 'B', 'B', 'A', 'B', 'A',
'A', 'A', 'A', 'A', 'B', 'B', 'A', 'A', 'A', 'A', 'A', 'A', 'B', 'B', 'A', 'B',
'B', 'A', 'B', 'A', 'B', 'A', 'B', 'B', 'A', 'A', 'B', 'A', 'A', 'B', 'A', 'A',
'A', 'A', 'B', 'A', 'A', 'A', 'B', 'A', 'B', 'A', 'B', 'A', 'A', 'A', 'B', 'A',
'A', 'A', 'B', 'B', 'A', 'B', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'B', 'A', 'A',
'A', 'B', 'A', 'A', 'B', 'A', 'A', 'A', 'A', 'B', 'A', 'B', 'A', 'B', 'A', 'A',
'B', 'B', 'B', 'B', 'A', 'B', 'B', 'A', 'A', 'A', 'B', 'B', 'B', 'B', 'A', 'B',
'A', 'A', 'A', 'B', 'A', 'B', 'A', 'A', 'B', 'B', 'A', 'B', 'B', 'B', 'B', 'B',
'B', 'B', 'A', 'B', 'B', 'B', 'A', 'B', 'A', 'A', 'B', 'B', 'B', 'A', 'B', 'B',
'A', 'A', 'B', 'A']
```