**HMM TRAINING**

**1. INTRODUCTION**

**HMM training** is the process to learn HMM parameters.

**Learning** : Given an observation sequence **O** and the set of possible states in the HMM, find the HMM parameters **A** (transition probabilities), **B** (emission probabilities) and the **pi** (states start probabilities).

The standard algorithm for HMM training is the **forward-backward**, or **Baum-Welch algorithm**, a special case of the **Expectation-Maximization** or **EM** algorithm. The algorithm will let us train both the transition probabilities Aand the emission probabilities B of the HMM. EM is an iterative algorithm, computing an initial estimate for the probabilities, then using those estimates to computing a better estimate, and so on, iteratively improving the probabilities that it learns.

**2. IMPLEMENTATION**

**gamma**: Probability of being in state j at time t

**zi** : Probability of being in state i at time t and state j at time t + 1

**alpha\_prob** : forward probability (likelihood)

**alpha** : forward matrix

**beta\_prob** : backward probability

**beta** : backward matrix

**A** : transition probabilities

**B** : emission probabilities

*1. Initialization :*

- gamma=[{} for t in range(len(obs\_sequence))] :

- zi = [{} for t in range(len(obs\_sequence) - 1)] :

- Computes alpha\_prob and alpha by using forward algorithm

- Computes beta\_prob and beta by using backward algorithm

*2. Compute gamma, pi and zi values :*

**Iterate from t=0 … len(obs\_sequence)**

**Iterate for s in states**

Compute gamma[t][s] = (alpha[t][s] \* beta[t][s]) / alpha\_prob

if t==0, update pi[s] = gamma[t][s]

**Iterate for s1 in states**

zi[t][s][s1] =alpha[t][s] \* A[s][s1] \* B[s1][obs\_sequence[t + 1]] \* beta[t + 1][s1] /alpha\_prob

*3. Update transition probabilities :*

**Iterate for s in states**

**Iterate for s1 in states**

zi\_sum = sum((zi[t][s][s1]) for t in range(len(obs\_sequence) – 1))

gamma\_sum = sum ((gamma[t][s]) for t in range(len(obs\_sequence) – 1))

A[s][s1] = zi\_sum / gamma\_sum

*4. Update emission probabilities :*

**Iterate for s in states**

**Iterate for obs in obs\_sequence**

gamma\_sum = 0.0

**Iterate from t=0 … len(obs\_sequence)**

if obs\_sequence[t]==obs, gamma\_sum += gamma[t][s]

emit\_gamma\_sum = sum ((gamma[t][s]) for t in range(len(obs\_sequence)))

B[s][obs] = gamma\_sum / emit\_gamma\_sum

*5. Return* A, B, pi