**HMM TRAINING**

**1. INTRADUCTION**

**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** : forward probability (likelihood)

**fwd** : forward matrix

**beta** : backward probability

**bwd** : backward matrix

**A** : transition probabilities

**B** : emission probabilities

*Initialization :*

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

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

- Computes alpha and fwd by using forward algorithm

- Computes beta and bwd by using backward algorithm

*Compute gamma, pi and zi values :*

**- Iterate from t=0 … len(obs)**

**- Iterate for s in states**

- Compute gamma[t][s] = (fwd[t][s] \* bwd[t][s]) / alpha

- If t==0, compute pi = gamma[t][s]

**-** **Iterate for s1 in states**

- zi[t][s][s1] =fwd[t][s] \* A[s][s1] \* B[s1][obs[t + 1]] \* bwd[t + 1][s1] /alpha

*Re-estimate :*

**- Iterate for s in states**

**-** **Iterate for s1 in states**

- A[s][s1] = sum((zi[t][s][s1]) for t in range(len(obs) – 1))

/ sum ((gamma[t][s]) for t in range(len(obs) – 1))

*Re-estimate gamma :*

**- Iterate for s in states**

**- Iterate for o in obs**

-val = 0.0

**- Iterate from t=0 … len(obs)**

- If obs[t]==o, val += gamma[t][s]

- B[s][o] = val / sum ((gamma[t][s]) for t in range(len(obs)))

*Return* A, B, pi