# **Assignment Report: Hidden Markov Model for Speech Recognition**

## **1. Introduction**

This report outlines the implementation of a speech recognizer using a Hidden Markov Model (HMM). The model was developed to process precomputed Mel-Frequency Cepstral Coefficients (MFCCs) for isolated word recognition. The implementation involves feature normalization, initialization of HMM parameters, decoding using the Viterbi algorithm, and evaluation metrics, including recognition error rate and a confusion matrix.

The primary goal of this assignment is to implement a speech recognizer based on a **Hidden Markov Model (HMM)** that can identify isolated words from a small vocabulary using **Mel-Frequency Cepstral Coefficients (MFCCs)** as features. The system aims to process precomputed MFCCs, apply normalization, initialize HMM parameters, and use the Viterbi algorithm to decode the sequence of states. Performance is evaluated using the **Recognition Error Rate (RER)** and a **Confusion Matrix**

### **Key Objectives:**

* **Feature Extraction**: Properly extract and normalize MFCC features.
* **HMM Initialization**: Correctly initialize HMM parameters (transition matrix A, emission probabilities B, and initial state probabilities π).
* **Decoding**: Use the Viterbi algorithm to decode the state sequence.
* **Evaluation**: Compute the recognition error rate and generate a confusion matrix.

## **Implementation Details**

## **Methodology**

### **2.1 Feature Extraction**

MFCCs are commonly used in speech recognition systems as they capture the spectral properties of speech signals. The extraction process involves:

1. **Pre-emphasis**: This process boosts the high-frequency components of the speech signal to compensate for the loss of high frequencies during recording.
2. **Frame and Hop Size**: The audio signal is divided into frames (30ms) with a hop size of 10ms (20ms overlap). This segmentation ensures the features are computed at regular intervals.
3. **MFCC Calculation**: The MFCCs are extracted from each frame of the signal. In this implementation, we use 13 coefficients, which are sufficient for recognizing isolated words.

**Normalization** of the MFCC features is essential to ensure that each feature dimension (corresponding to a different frequency band) has the same scale. This prevents certain features from dominating the model due to their larger magnitudes.

### **2.2 Hidden Markov Model Initialization**

The **Hidden Markov Model (HMM)** is a probabilistic model where the system is assumed to be in one of a set of hidden states at any given time. It involves the following components:

* **State Transition Matrix (AAA)**: This matrix models the probabilities of transitioning from one state to another. For simplicity, we use a **left-right HMM**, meaning that the model is constrained to move from state 1 to state N in a left-to-right fashion. The transition probabilities are initialized as follows:
  + Self-loop probability: 0.8 (i.e., the model has a high chance of staying in the same state).
  + Transition to the next state: 0.2 (i.e., the model has a small chance of moving to the next state).
  + The last state has a self-loop with a probability of 1.0, meaning the model stays in the last state.
* **Emission Matrix (B)**: This matrix represents the likelihood of observing a particular feature vector given a specific state. Initially, it is randomized, and the features are normalized.
* **Initial State Probabilities (π)**: The initial state probabilities are set so that the first state has a probability of 1, and all other states have zero probability.

### **2.3 Viterbi Algorithm**

The **Viterbi algorithm** is used for **decoding** the sequence of hidden states given the observation sequence. The algorithm uses dynamic programming to compute the maximum likelihood of the entire sequence by recursively calculating the most probable state at each time step.

#### **Steps of the Viterbi Algorithm:**

1. **Initialization**: At time t=1, the probability of each state is computed using the initial state probabilities (π) and the emission probabilities (B).
2. **Recursion**: For each subsequent time step t, the algorithm computes the maximum probability of each state by considering all transitions from previous states. This is done by combining the previous state’s probability and the transition probability.
3. **Termination**: After processing all time frames, the algorithm selects the state with the highest probability at the final time frame.
4. **Backtracking**: The state sequence is traced back using the back pointer matrix, which keeps track of the state transitions that led to the optimal path.

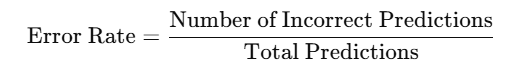
#### **Numerical Stability:**

* To avoid **underflow** issues, probabilities are represented in logarithmic space. This ensures that the product of small probabilities does not result in numerical errors. Instead of multiplying probabilities, we add their logarithms.

### **2.4 Evaluation**

To assess the performance of the speech recognizer, two evaluation metrics are used:

1. **Recognition Error Rate (RER)**: This is the percentage of misclassified predictions in the test set compared to the ground truth. The formula for RER is:



1. **Confusion Matrix**: A confusion matrix is a table used to describe the performance of a classification model. Each row represents the true labels, while each column represents the predicted labels. It helps visualize the types of misclassifications made by the model.