Speech Signal Analysis

DEF (*Dithering*) Dithering adds noise to the signal which avoids the signal having zeros (which may be problematic when taking the log).

$$y[t] = x[t] + \epsilon[t] \qquad \epsilon[t] \sim \mathcal{N}(0, \sigma^2)$$

DEF (*DC Offset*) Most processing techniques assume that the signal is centered around 0.

$$y[t] = x[t] - \frac{1}{T}\mu_x \qquad \mu_x = \frac{1}{T}\sum_{t=1}^{T} x[t]$$

DEF (*Pre-emphasis*) Emphasize high frequency components.

$$y[t] = x[t] - \alpha x[t-1]$$

where α is a constant (usually 0.97)

DEF (*Discrete Fourier Transform (DFT)*) The dot product of the signal with the sinusoids of the frequency (Fourier basis).

$$X[k] = \sum_{t=0}^{T-1} x[t] e^{i2\pi kt/T}$$

DFT decomposes the signal into frequency components. Commonly $\mathcal{F}$ is used to denote the DFT operator.

THM (*Properties of DFT*)

- Linearity: $\mathcal{F}(a_1 x_1 + a_2 x_2) = a_1 \mathcal{F}(x_1) + a_2 \mathcal{F}(x_2)$
- Shift: If $y[t] = x[t-m]$, then $Y[k] = e^{i2\pi km/T} X[k]$

DEF (*Windowing*) Given a signal $x[t]$, a window function $w[t]$ (Hamming, Hann, etc.)

$$y[t] = x[t]w[t]$$

DEF (*Short-Time Fourier Transform (STFT)*) Given a signal $x[t]$, a window function $w[t]$, and a hop size h, the STFT is given by:

$$X[k, m] = \sum_{t=0}^{T-1} x[t] w[t-m] e^{-i2\pi kt/T}$$

where m is the current frame index and k is the frequency index. This gives a complex spectrogram of the signal.

DEF (*Complex Spectrogram*) Given a frequency bin $X[k] = a + bi$,

- Real: $\mathcal{R}(X[k]) = a$
- Imaginary: $\Im m(X[k]) = b$
- Magnitude: $|X[k]| = \sqrt{a^2 + b^2}$
- Phase: $\angle X[k] = \arccos \frac{a}{\sqrt{a^2+b^2}}$
- Energy: $P[k] = |X[k]|^2$

DEF (*Mel Spectrogram*) After obtaining the spectrogram, $X[k]$, and using the mel filter bank, $H_n[k]$, the mel spectrogram is given by:

$$Y[k, n] = \sum_{k=0}^{K-1} |X[k]| H_n[k]$$

DEF (*Mel Frequency Cepstral Coefficients (MFCCs)*) A common feature extraction technique for speech recognition which is able to capture speaker characteristics and phonetic content.

1. Extract the mel spectrogram $Y[k, n]$
2. Apply the Discrete Cosine Transform (DCT) to the mel spectrogram.
3. Take the first p coefficients (usually 13).

Hidden Markov Models

DEF (*Objective of ASR*) Given a sequence of acoustic feature vectors X, and W denotes a word sequence, ASR aims to find the most likely word sequence W^*

$$W^* = \arg\max_W P(W|X)$$

COR (*Decomposition of $P(W|X)$*) We can decompose $P(W|X)$ with Bayes' theorem:

$$P(W|X) = \frac{P(X|W)P(W)}{P(X)} \propto P(X|W)P(W)$$

$P(X|W)$ is the **Acoustic Model** and $P(W)$ is the **Language Model**.

DEF (*Modelling the Acoustic Model with HMMs*) Commonly, the left-to-right HMM is used to model the acoustic model. As a word is composed of multiple phonemes, we can model each phoneme with a left-to-right HMM and then concatenate them to form a HMM for the word. For the phoneme HMMs, we typically use a left-to-right HMM with 3 states which as a consequence also enforces a minimum phone duration.

DEF (*The three fundamental problems of HMMs*)

1. **Likelihood** - Determine the overall likelihood of an observation sequence $X = x_1, x_2, \ldots, x_T$ given an HMM topology $\mathcal{M}$:
$$P(X|\mathcal{M})$$
2. **Decoding** - Determine the most likely sequence of hidden states $Q = q_1, q_2, \ldots, q_T$ given an observation sequence $X = x_1, x_2, \ldots, x_T$ and an HMM topology $\mathcal{M}$
3. **Training** - Given an observation sequence $X = x_1, x_2, \ldots, x_T$ and an HMM topology $\mathcal{M}$, determine the optimal state occupation probabilities.

ALG (*Forward Probability [$\alpha_j(t)$]*) The **Likelihood** and **Training** problem rely on forward probabilities. The forward probability $\alpha_j(t)$ is the probability of observing the observation sequence $x_1, \cdots, x_t$ and being in state j at time t. This can be computed recursively:

- Initialisation:
$$\alpha_j(0) = 1 \quad j = 0$$
$$\alpha_j(0) = 0 \quad j \neq 0$$
- Recursion:
$$\alpha_j(t) = \left(\sum_{i=0}^{J} \alpha_i(t-1) a_{ij}\right) b_j(x_t)$$
- Termination:
$$P(X|\mathcal{M}) = \alpha_E = \sum_{i=1}^{J} \alpha_i(T) a_{iE}$$

ALG (*Viterbi Algorithm*) The **Decoding** problem is solved by the **Viterbi Algorithm**. Define likelihood of the most probable partial path in state j at time t as $V_j(t)$ and the Viterbi backpointer as $B_j(t)$ (the most probable state at time $t-1$ that leads to state j at time t). Then the Viterbi algorithm can be described as:

- Initialisation:
$$V_0(0) = 1$$
$$V_j(0) = 0 \quad j \neq 0$$
$$B_j(0) = 0$$
- Recursion:
$$V_j(t) = \max_{i=0}^{J} V_i(t-1) a_{ij} b_j(x_t)$$
$$B_j(t) = \arg\max_{i=0}^{J} V_i(t-1) a_{ij}$$

- Termination:

$$V_E = \max_{j=1}^{J} V_j(T) a_{jE}$$
$$B_E = \arg\max_{j=1}^{J} V_j(T) a_{jE}$$

ALG (*Training: Hard Assignment with Viterbi Training*) If we know the state-time alignment of the training data (i.e. the sequence of states that generated the observation sequence), then the transition and observation probabilities can be estimated as follows:

$$a_{ij} = \frac{C(i \to j)}{\sum_k C(i \to k)}$$
$$\mu_j = \frac{\sum_t z_{jt} x_t}{\sum_t z_{jt}}$$
$$b_j = \mathcal{N}(x_t | \mu_j, \Sigma_j)$$

where z_{jt} is an indicator variable that is 1 if the t-th observation was generated by the j-th state. $C(i \to j)$ is the number of times the transition from state i to state j is made.

DEF (*Backward Probability [$\beta_j(t)$]*) The backward probability $\beta_j(t) = P(x_{t+1}, \ldots, x_T | q_t = j, \mathcal{M})$ is the probability of observing the observation sequence from time $t+1$ to the end, given that the HMM is in state j at time t.

- Initialisation:

$$\beta_j(T) = a_{jE}$$

- Recursion:

$$\beta_i(t) = \sum_{j=1}^{J} a_{ij} b_j(x_{t+1}) \beta_j(t+1)$$

- Termination:

$$P(X|\mathcal{M}) = \beta_0(0) = \sum_{j=1}^{J} a_{0j} b_j(x_1) \beta_j(1) = \alpha_E$$

DEF (*State Occupation Probability [$\gamma_j(t)$]*) $\gamma_j(t)$ is the probability of being in state j at time t, given the complete observation sequence X and the model $\mathcal{M}$. It combines forward and backward probabilities.

$$\begin{aligned}
\gamma_j(t) &= P(q_t = j | X, \mathcal{M}) \\
&= \frac{P(q_t = j, X_{1:T} | \mathcal{M})}{P(X|M)} \\
&= \frac{P(q_t = j, X_{1:t}, X_{t+1:T} | \mathcal{M})}{\alpha_E} \\
&= \frac{P(X_{t+1:T} | q_t = j, X_{1:t}, \mathcal{M}) \times P(q_t = j, X_{1:t} | \mathcal{M})}{\alpha_E} \\
&= \frac{P(X_{t+1:T} | q_t = j, \mathcal{M}) \times P(q_t = j, X_{1:t} | \mathcal{M})}{\alpha_E} \\
&= \frac{\beta_j(t) \alpha_j(t)}{\alpha_E}
\end{aligned}$$

This represents the expected proportion of time spent in state j at time t. Note that $\sum_{j=1}^{J} \gamma_j(t) = 1$ for any t.

DEF (*Transition Occupation Probability ($\xi_{ij}(t)$)*) $\xi_{ij}(t)$ is the probability of being in state i at time t and transitioning to state j at time $t+1$, given the complete observation sequence X and the model $\mathcal{M}$.

$$\begin{aligned}
\xi_{ij}(t) &= P(q_t = i, q_{t+1} = j | X, \mathcal{M}) \\
&= \frac{P(q_t = i, q_{t+1} = j, X_{1:T} | \mathcal{M})}{P(X|\mathcal{M})} \\
&= \frac{P(X_{1:t}, q_t = i, q_{t+1} = j, X_{t+1:T} | \mathcal{M})}{\alpha_E} \\
&= \frac{\alpha_i(t) a_{ij} b_j(x_{t+1}) \beta_j(t+1)}{\alpha_E}
\end{aligned}$$

ALG (*EM Algorithm (Baum-Welch) for HMM Training*) The Baum-Welch algorithm is an instance of the Expectation-Maximization (EM) algorithm used to find the maximum likelihood estimate of the parameters ($\lambda = \{A, B, \pi\}$) of an HMM when the state sequence is unknown. It iteratively performs two steps:

- **E-step (Expectation):** Using the current parameter estimates λ^{old}, compute the expected values needed for the M-step. This involves:
  - Computing forward probabilities $\alpha_j(t)\ \forall j, t$.
  - Computing backward probabilities $\beta_j(t)\ \forall j, t$.
  - Computing state occupation probabilities $\gamma_j(t)\ \forall j, t$
  - Computing transition occupation probabilities $\xi_{ij}(t)\ \forall i, j, t$
  - If using GMMs for b_j, compute component-state occupation probabilities $\gamma_{jm}(t)$ (see below).
- **M-step (Maximization):** Update the parameters λ^{new} to maximize the expected complete-data log-likelihood, using the expectations computed in the E-step. See specific update formulas below.
- Repeat E-step and M-step until the parameters or the likelihood $P(X|\lambda)$ converge.

DEF (*M-Step: Transition Probabilities (a_{ij})*) The transition probability a_{ij} is updated as the expected number of transitions from state i to j, divided by the expected total number of transitions *out* of state i.

$$\begin{aligned}
\hat{a}_{ij}^{\text{new}} &= \frac{\text{Expected transitions } i \to j}{\text{Expected transitions out of } i} \\
&= \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{k=1}^{J} \sum_{t=1}^{T-1} \xi_{ik}(t)} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}
\end{aligned}$$

For a corpus of R utterances, sum the numerators and denominators over all utterances $r = 1, \ldots, R$.

DEF (*M-Step: Observation Probabilities ($b_j(\mathbf{x})$) [Single Gaussian]*) If the observation probability for state j is modeled by a single multivariate Gaussian $b_j(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, the parameters are updated using weighted maximum likelihood, where the weights are the state occupation probabilities $\gamma_j(t)$.

$$\hat{\boldsymbol{\mu}}_j^{\text{new}} = \frac{\sum_{t=1}^{T} \gamma_j(t) \mathbf{x}_t}{\sum_{t=1}^{T} \gamma_j(t)}$$
$$\hat{\boldsymbol{\Sigma}}_j^{\text{new}} = \frac{\sum_{t=1}^{T} \gamma_j(t) (\mathbf{x}_t - \hat{\boldsymbol{\mu}}_j^{\text{new}})(\mathbf{x}_t - \hat{\boldsymbol{\mu}}_j^{\text{new}})^T}{\sum_{t=1}^{T} \gamma_j(t)}$$

DEF (*M-Step: Observation Probabilities ($b_j(\mathbf{x})$) [GMM]*) If $b_j(\mathbf{x})$ is an M-component GMM, $b_j(\mathbf{x}) = \sum_{m=1}^{M} c_{jm} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm})$, we need the component-state occupation probability $\gamma_{jm}(t)$.

- **Component-State Occupation Probability ($\gamma_{jm}(t)$):** This is the probability of being in state j and using component m at time t, given X. It's calculated in the E-step after $\gamma_j(t)$:

$$\begin{aligned}
\gamma_{jm}(t) &= P(q_t = j, c_t = m | X, \mathcal{M}^{\text{old}}) \\
&= \gamma_j(t) \times \frac{c_{jm}^{\text{old}} \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{jm}^{\text{old}}, \boldsymbol{\Sigma}_{jm}^{\text{old}})}{b_j^{\text{old}}(\mathbf{x}_t)} \\
&= \frac{\alpha_j(t) \beta_j(t)}{\alpha_E} \times \frac{c_{jm}^{\text{old}} \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{jm}^{\text{old}}, \boldsymbol{\Sigma}_{jm}^{\text{old}})}{\sum_{k=1}^{M} c_{jk}^{\text{old}} \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{jk}^{\text{old}}, \boldsymbol{\Sigma}_{jk}^{\text{old}})}
\end{aligned}$$

- **GMM Parameter Updates (M-step):** Use $\gamma_{jm}(t)$ as weights for component m of state j.

$$\hat{c}_{jm}^{\text{new}} = \frac{\sum_{t=1}^{T} \gamma_{jm}(t)}{\sum_{k=1}^{M} \sum_{t=1}^{T} \gamma_{jk}(t)} = \frac{\sum_{t=1}^{T} \gamma_{jm}(t)}{\sum_{t=1}^{T} \gamma_j(t)}$$
$$\hat{\boldsymbol{\mu}}_{jm}^{\text{new}} = \frac{\sum_{t=1}^{T} \gamma_{jm}(t) \mathbf{x}_t}{\sum_{t=1}^{T} \gamma_{jm}(t)}$$
$$\hat{\boldsymbol{\Sigma}}_{jm}^{\text{new}} = \frac{\sum_{t=1}^{T} \gamma_{jm}(t) (\mathbf{x}_t - \hat{\boldsymbol{\mu}}_{jm}^{\text{new}})(\mathbf{x}_t - \hat{\boldsymbol{\mu}}_{jm}^{\text{new}})^T}{\sum_{t=1}^{T} \gamma_{jm}(t)}$$