Lab3 Report

Name: WU YIHANG.   StuID: A0285643W

E-STEP

In my implementation, this algorithm is divided into 2 parts.

- The first part is the forward part, where I compute $\tilde{\alpha}(z_n), p(x_n|z_n), c_n$(the scaling factor), based on these two equations below in the lecture's slides:

$$c_n \widehat{\alpha}(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \widehat{\alpha}(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1}) = \tilde{\alpha}(\mathbf{z}_n) \qquad c_n = \sum_{z_n} \tilde{\alpha}(\mathbf{z}_n)$$

  - $p(x_n|z_n)$: I calculate this using the `scipy.stats.norm.pdf` function from the SciPy library
  - $\tilde{\alpha}(z_n)$ are calculated before c_n, α in each loop
    - For $\tilde{\alpha}(z_0)$, it is initialized according to $\alpha(\mathbf{z}_1) = p(\mathbf{x}_1, \mathbf{z}_1) = p(\mathbf{z}_1)p(\mathbf{x}_1|\mathbf{z}_1) = \prod_{k=1}^{K} \{\pi_k p(\mathbf{x}_1|\boldsymbol{\phi}_k)\}^{z_{1k}}$
    - Then, $\tilde{\alpha}(z_n)$ can be calculated according to the first equation above.
  - Then we can get c_n using `np.sum(...)`, after which we can get the final $\alpha(z_n) = \tilde{\alpha}(z_n)/c_n$
- The second part is the backward part, which is easier than the first part. We only need to compute the β list according to the equation below: $c_{n+1}\widehat{\beta}(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} \widehat{\beta}(\mathbf{z}_{n+1}) p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1}) p(\mathbf{z}_{n+1}|\mathbf{z}_n) = \tilde{\beta}(\mathbf{z}_n)$
  - First, all $\beta(z_n)$ is initialized by 1
  - And then $\tilde{\beta}(z_n)$ is calculated accoring to this equation.
  - Then we can get the final $\beta(z_n) = \tilde{\beta}(z_n)/c_{n+1}$
- Finally, we can get the $\gamma(z_n), \xi(z_{n-1}, z_n)$, according to the equation:

$$\begin{aligned} \gamma(\mathbf{z}_n) &= \widehat{\alpha}(\mathbf{z}_n)\widehat{\beta}(\mathbf{z}_n) \\ \xi(\mathbf{z}_{n-1}, \mathbf{z}_n) &= c_n^{-1} \widehat{\alpha}(\mathbf{z}_{n-1}) p(\mathbf{x}_n|\mathbf{z}_n) p(z_n|z_{n-1}) \widehat{\beta}(\mathbf{z}_n) \end{aligned}$$

M-Step

In this algorithm, we update those parameters, like $A, \pi_k, \mu_k, \Sigma_k$, accoring to the equation in lecture's slides: $\pi_k = \frac{\gamma(z_{1k})}{\sum_{j=1}^{K} \gamma(z_{1j})}$, $A_{jk} = \frac{\sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nk})}{\sum_{l=1}^{K} \sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nl})}$. $\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^{N} \gamma(z_{nk})}$, $\boldsymbol{\Sigma}_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}}{\sum_{n=1}^{N} \gamma(z_{nk})}$.

As the variables we need have been calculated in E-step, this step we only need to do some matrix operation to update out parameter.

PUTTING THEM TOGETHER

In this algorithm, we do iterations, and identify when we can stop the loop. In my implementation, a threshold of 1e-4 is set, and the old value of μ, σ are stored, and then I calculate $abs(value_{new} - value_{new})$ for both new μ and σ. It their results are both less than the threshold, we stop the iteration and get out final estimation.