**DS-GA 3001.001 Special Topics in Data Science: Modeling Time Series**
**Homework 3 solution**

**Pseudocode for alpha-beta algorithm**

For marginal inference we use the scaled versions of $\alpha$ and $\beta$:

$$\hat{\alpha}(\mathbf{z}_i) = \mathrm{P}\left(\mathbf{z}_i|\mathbf{x}_{1:i}\right) = \frac{\alpha(\mathbf{z}_i)}{\mathrm{P}\left(\mathbf{x}_{1:i}\right)} \tag{1}$$

$$\hat{\beta}(\mathbf{z}_i) = \frac{\mathrm{P}\left(\mathbf{x}_{i+1:t}|\mathbf{z}_i\right)}{\mathrm{P}\left(\mathbf{x}_{\mathbf{i+1:t}}|\mathbf{x}_{1:i}\right)} = \frac{\beta(\mathbf{z}_i)}{\prod_{j=i+1:t} c_j} \tag{2}$$

where the scaling factors are defined using an intermediate quantity, $c_i$:

$$c_i = \mathrm{P}\left(\mathbf{x}_i|\mathbf{x}_{1:i-1}\right) \tag{3}$$

$$\mathrm{P}\left(\mathbf{x}_{1:i}\right) = \prod_{j=1:i} c_j \tag{4}$$

The new expressions for the posterior marginals are to compute become:

$$\gamma(\mathbf{z}_i) = \hat{\alpha}(\mathbf{z}_i)\hat{\beta}(\mathbf{z}_i) \tag{5}$$

$$\xi(z_{i,j}, z_{i+1,k}) = {c_{i+1}}^{-1}\hat{\alpha}(\mathbf{z}_i)\mathrm{P}\left(\mathbf{z}_{i+1}|\mathbf{z}_i\right)\mathrm{P}\left(\mathbf{x}_{i+1}|\mathbf{z}_{i+1}\right)\hat{\beta}(\mathbf{z}_{i+1}) \tag{6}$$

The scaling translates into updated recursion equations:

$$c_i\hat{\alpha}(\mathbf{z}_i) = \mathrm{P}\left(\mathbf{x}_i|\mathbf{z}_i\right)\sum_{\mathbf{z}_{i-1}}\hat{\alpha}(\mathbf{z}_{i-1})\mathrm{P}\left(\mathbf{z}_i|\mathbf{z}_{i-1}\right) \tag{7}$$

$$c_{i+1}\hat{\beta}(\mathbf{z}_i) = \sum_{\mathbf{z}_{i+1}}\hat{\beta}(\mathbf{z}_{i+1})\mathrm{P}\left(\mathbf{x}_{i+1}|\mathbf{z}_{i+1}\right)\mathrm{P}\left(\mathbf{z}_{i+1}|\mathbf{z}_i\right) \tag{8}$$

with the added complication that we need to compute and keep track of $c_i$ values. In practice, this means that we first finish the $\hat{\alpha}$ update sweep: on each step we first compute the unnormalized probability

$$\mathbf{q} = \mathrm{P}\left(\mathbf{x}_i|\mathbf{z}_i\right)\sum_{\mathbf{z}_{i-1}}\hat{\alpha}(\mathbf{z}_{i-1})\mathrm{P}\left(\mathbf{z}_i|\mathbf{z}_{i-1}\right),$$

then its corresponding normalizing constant which gets saved as $c_i = \sum_k q_k$, with $\hat{alpha}_i = \frac{1}{c_i}\mathbf{q}$. The corresponding $\beta$s get normalized using these saved values (hence the backward pass can only be computed after the forward pass has finished).

The results of the comparison to Viterbi will depend on the exact parameters, but the more uncertainty in the posterior over latents the more likely are the two outputs to differ. This can be seen by increasing the overlap of the individual components (e.g. moving them closer together or increasing the noise variance), also by increasing the dimensionality of the state space (increasing $K$).

To test validity of the solution: a) make the observation noise very small check latents inferred correctly. b) same observation noise for all classes, but transition probability highly structured, check that posterior reflects transitions in $\mathbf{A}$, as the only available source of information. Anything else people came up with as long as it's appropriately justified.

**Pseudocode for EM**

For the parameter initialization, one could start with a random guess. A more elegant solution, with faster convergence, is to first fit a traditional gaussian mixture to the data (ignoring parameter dependencies) by EM (ok to do this usinf library; bonus points if implemented code). This will result in parameter estimates for the mean and covariance of each component, $\mu_k$ and $\Sigma_k$ and some prior over classes $\pi$ which can be used as proxy for $p(x_0)$. To estimate the transition probabilities, $\mathbf{A}$, one can use the posterior over classes for each observation and do some (possibly weighted) counting.

Once the initialization is in place, we alternate between the E steps, which involve running the alpha-beta algorithm with the current estimates, computing $\gamma(z_{i,k})$ and $\xi(\mathbf{z}_i, \mathbf{z}_{i+1})$. In the M-step we update the parameters as:

$$\pi_k^{\text{new}} = \frac{\gamma(z_{1,k})}{\sum_j \gamma(z_{1,j})} \tag{9}$$

$$A_{jk}^{\text{new}} = \frac{\sum_i \xi(z_{i,j}, z_{i+1,k})}{\sum_{i,l} \xi(z_{i,j}, z_{i+1,l})} \tag{10}$$

$$\mu_k^{\text{new}} = \frac{1}{\sum_i \gamma(z_{i,k})} \sum_i \gamma(z_{i,k}) \mathbf{x}_i \tag{11}$$

$$\Sigma_k^{\text{new}} = \frac{1}{\sum_i \gamma(z_{i,k})} \sum_i \gamma(z_{i,k}) \mathbf{x}_i \mathbf{x}_i^{\text{t}} - \mu_k \mu_k^{\text{t}} \tag{12}$$

The results for the $K = 3$ case will reproduce those in tsa4. For $K = 2$ the fit will be worse, but one should still be able to identify the high variance component with reasonable accuracy.