A simplified form of Speech Recognition with Markov models

1 - Language detection

There are three languages: A, B, and C. Each language uses the same set of symbols: "A, o, e, t, p, g, and k. However, each language uses the symbols differently. In each of these languages we can model everything as P(next symbol | current symbol).

There is training data available for each language. This consists of several files each generated by sampling from a Markov model. Using python, I built a Markov model for each of the languages. I then used the Markov model and Bayes' rule to classify the test cases.

We see that there are 7 observed states, corresponding to the set of symbols "A, o, e, t, p, g, and k". First, I train the data available for each language by building the transition matrix, which contains the probabilities of transition from state $s_i = a$ to the next state $s_{i+1} = b$, using the formula:

$$p(s_i = a, s_{i+1} = b) = \frac{count(s_i=a,\, s_{i+1}=b)}{count(s_i=a)}$$

After building the 3 transition matrices for three languages using the training data for each language, I use the matrices to calculate the likelihood of observing the test string, given each language. Because the likelihood gets super small with some computations, I calculated the log likelihood and converted back to the likelihood in the end. The formula looks like:

$$p(string|language\ k) = p_k(x_1)p_k(x_2|x_1)p_k(x_3|x_2)...p_k(x_n|x_{n-1})$$

with n: the length of test string

and the probability of transition from this state to the next state, taken from each transition matrix

After we got the likelihood, we use the Bayes formula to calculate the posterior - the probability of it being a specific language given the observed string in the test data:

$$p(language\ k|string) = \frac{p(string|language\ k) * p(language\ k)}{p(string)}$$

In this case we have only 3 languages so

$$p(string) = p(string|lang\ A) * p(lang\ A) + p(string|lang\ B) * p(langB) + p(string|lang\ C) * p(lang\ C)$$

Assuming that we have uninformed prior, I compared the posterior of three languages which are proportional to the likelihood divided by the evidence. From there, I got the posterior for each test case of the test data.

2 - Speaker identification

There are three people in a room. Each says about 10 phonemes, before being randomly interrupted by someone else. When they speak they all sound the same, however each person tends to use different phonemes in their speech. Specifically we can model the following transition probabilities that someone will interrupt the current speaker: P(speaker i at time t+1 | speaker j at time t). We can also model the probability over phonemes given a particular speaker: P(phoneme | speaker i). The phonemes are identical to the ones introduced in problem 1 (but the transition matrices are obviously different, since they take a different form altogether).

1/ The update equations that are needed to train a hidden Markov model and the initialization for the transition matrix:

I created the initial emission matrix by sampling from a dirichlet distribution the distribution of probability that each speaker will speak a certain phoneme. The benefit is that the dirichlet distribution automatically sums to 1, so we don't have to normalize. More impor, by changing the parameters of the distribution we can change the "randomness" of individual numbers. Specifically, depending on the main parameter the Dirichlet distribution will either give vectors where all the values are close to 1./N where N is the number of phoneme, or give vectors where most of the values of the vectors will be ~0 , and there will be a single 1, or give something in between those possibilities. Because we will use this for EM, we don't want our initial state to be so "symmetric", which is the case where all the values are close to 1/.N, instead, I chose to make the values a little bit unbalanced. The emission matrix looks something like:

	A	o	e	t	p	g	k
Speaker 1	0.13428	0.21190	0.1558	0.1328	0.0929	0.17516	0.0971
Speaker 2	0.18873	0.28137	0.04105	0.133168	0.063887	0.07125	0.22054
Speaker 3	0.00787	0.00974	0.00218	0.1909	0.160804	0.217888	0.41064

For the transition matrix, as you suggested on Slack, due to the information that each speaker says about 10 phonemes, before being randomly interrupted by someone else, given that this phoneme is spoken by this person, the probability that the next phoneme is spoken by the same person would be 0.9 and the probability that it is spoken by each of the other two is (1-0.9)/2 = 0.05. From there, we got the initial transition matrix:

	Speaker 1	Speaker 2	Speaker 3
Speaker 1	0.9	0.05	0.05
Speaker 2	0.05	0.9	0.05
Speaker 3	0.05	0.05	0.9

For the initial hidden state : P(H_1), let's assume that it would be uniform, because we have a random prior for emission matrix, and a symmetric matrix for the prior of transition matrix, so basically all the speakers are interchangeable in this sense. Therefore, I assigned P(H_1) to be (1/3, 1/3, 1/3)

The way I constructed the EM process is

E-Step:

The function for this step will take the inputs: emission table, transition table, the initial hidden state, the list of phonemes and the data.
What we observe here is the phonemes, but we can used the emission table to get the likelihood, which is p(phoneme | speaker)
As seen above, I assigned the prior for initial speaker to be = (s1,s2,s3) = (1/3,1/3,1/3)
From there, I could calculate the posterior for this speaker, using the formula
$p(speaker|phoneme) = \frac{p(phoneme|speaker)*p(speaker)}{p(phoneme)}$
To get the prior for the next speaker, I multiplied the posterior of this speaker by the probability of transition from this speaker to the next speaker (from the transition table). To be more specific, we have 3 speaker in total, so we will have 9 joint probabilities:

$$p(s_i = s1, s_{i+1} = s1) = p(s_i = s1) * p(s_{i+1} = s1|s_i = s1)$$
$$p(s_i = s1, s_{i+1} = s2) = p(s_i = s1) * p(s_{i+1} = s2|s_i = s1)$$
$$p(s_i = s1, s_{i+1} = s3) = p(s_i = s1) * p(s_{i+1} = s3|s_i = s1)$$
$$p(s_i = s2, s_{i+1} = s1) = p(s_i = s2) * p(s_{i+1} = s1|s_i = s2)$$
$$p(s_i = s2, s_{i+1} = s2) = p(s_i = s2) * p(s_{i+1} = s2|s_i = s2)$$
$$p(s_i = s2, s_{i+1} = s3) = p(s_i = s2) * p(s_{i+1} = s3|s_i = s2)$$
$$p(s_i = s3, s_{i+1} = s1) = p(s_i = s3) * p(s_{i+1} = s1|s_i = s3)$$
$$p(s_i = s3, s_{i+1} = s2) = p(s_i = s3) * p(s_{i+1} = s2|s_i = s3)$$
$$p(s_i = s3, s_{i+1} = s3) = p(s_i = s3) * p(s_{i+1} = s3|s_i = s3)$$

From this, we can get the marginal probabilities of the next speaker being one of the speaker by sum up the conditional probabilities:

$$p(s_{i+1} = s1) = p(s_i = s1, s_{i+1} = s1) + p(s_i = s2, s_{i+1} = s1) + p(s_i = s3, s_{i+1} = s1)$$
$$p(s_{i+1} = s2) = p(s_i = s1, s_{i+1} = s2) + p(s_i = s2, s_{i+1} = s2) + p(s_i = s3, s_{i+1} = s2)$$
$$p(s_{i+1} = s3) = p(s_i = s1, s_{i+1} = s3) + p(s_i = s2, s_{i+1} = s3) + p(s_i = s3, s_{i+1} = s3)$$

The output for the expectation function is the posterior probabilities of each phoneme is spoken by each person. We have 1000 phonemes in the list and 3 speakers, so my output has dimension (1000,3)

M-Step:
So from the E-step, we got the probabilities for the hidden states, we would use these to compute the new maximum likelihood estimates of our parameters, which in this case are the transition matrix and the emission matrix.
First, I updated each cell in the emission matrix by first sum up all the probabilities of that particular speaker speaking each phoneme over the entire test string, and then divided that by the sum of probabilities that this speaker actually speak. (the value of cell divided by the sum of the row so that the probabilities sum to 1)

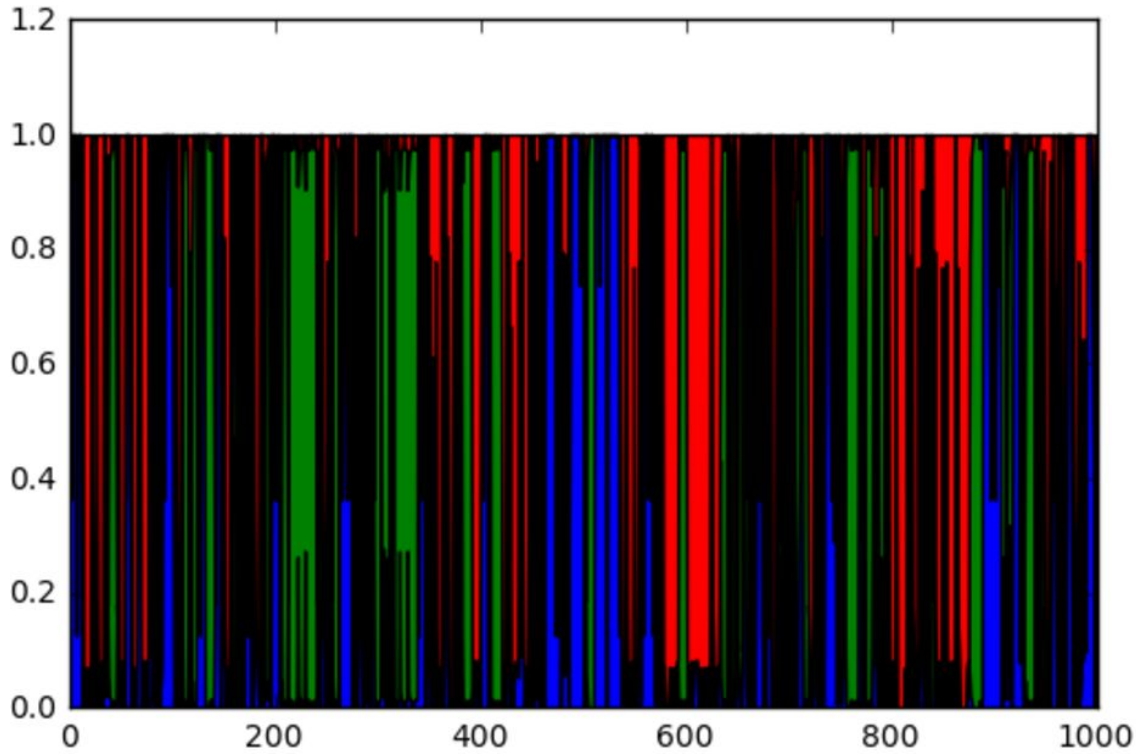
For the transition matrix, the value of cell ab would be the sum of the product of the probabilities of speaker b followed by speaker a, looping through the whole test string. Then to normalize, we divide the probabilities by the sum of each column (probability of this person being this particular person).

$$p(s_i = a, s_{i+1} = b) = \frac{sum(p(s_i=a, s_{i+1}=b))}{sum(p(s_i=a))}$$

The outputs for the maximization function are the two matrices: emission matrix and transition matrix

After I got the two functions, I performed the EM algorithms by iterate the process 1000 times, with the initial inputs as specified above. In each iteration, the expectation function used the parameters - emission table and transition table to compute the posterior probabilities - how likely the hidden states are. After that, the maximization function would use the updated probabilities to compute the new maximum likelihood estimates of our parameters.

2/ I used a stackplot from matplotlib to show the probability of a particular person speaking.



Taking the highest probability among the three to predict what person speaks that particular phoneme, I did a frequency table to see what the model predicts on the data:

Speaker	A	e	g	k	o	p	t	All
1	0	141	121	24	0	0	0	286
2	176	0	0	15	0	0	168	359
3	0	0	0	0	188	167	0	355
All	176	141	121	39	188	167	168	1000

We see that speaker 1 tends to speak phonemes “e”, ”g" and “k"; speaker 2 tends to speak phonemes “A”, “k" and “t" and speaker 3 tends to speak phonemes “o", “p".