Carnatic Music Classification using Hidden Markov Models

Ramchandran Muthukumar
Birla Institute of Technology and Science-Pilani, India
ramcha1994@gmail.com

November 2016

1. Introduction

Hidden Markov Models have successfully been used for many applications like speech, handwritten and gesture recognition. They have also been used in predicting nucleotide sequences in Biology and in regime-switching models in financial timeseries data. It is a probabilistic model and essentially assumes a structure of the environment in a certain form (subject to parameters like number of states). As with all probabilistic models, there is a lot of literature on estimation and inference and using these particular class of models. This project aims to model Carnatic Music using Hidden Markov Models. We train a HMM to capture a "Raga" and use this later to both classify and automatically generate musical sequences.

2. Hidden Markov Models

A stochastic process that satisfies markov property is called a Markov Chain. Markov Chains are heavily used in modelling many real-life scenarios where states of the system change with time (or in steps). Even in cases where the markov assumption needn't neccesarily hold , modelling the system using Markov Chains gives insightful results.

The one common drawback of modelling with Markov Chains is that it requires complete information of states. This might not be feasible and there are many cases where we can only partially observe the environment due to constraints.

This leads us to Hidden Markov Models (HMM) where we assume there is a hidden/latent layer which is not observable. The trellis diagram below is a pictorial description of the markov chain for HMM.

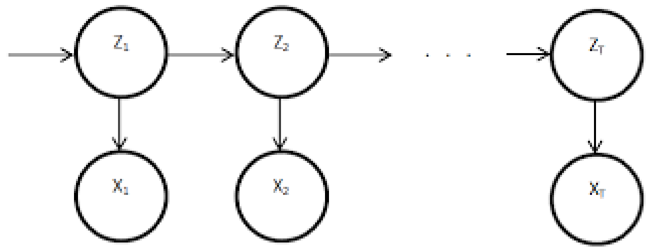

Figure 1: *Trellis Diagram*

$Z_1, Z_2, \ldots Z_n \in \{1, 2, \cdot, m\}-$ Hidden Random Variables.
$X_1, X_2, \ldots X_n \in \mathcal{X}-$ Observed Random Variables.

The set $\{1, 2, \cdot, m\}$ is the set of possible states. We encode the states by indexing them. They can mean anything like weather of the day, vowel/consonant etc. It is also referred to as the State Domain.

The random variable Z_i represents the state of the sequence at time/step i.

The set $\mathcal{X}$ is the set of all possible observations. It is also referred to as the Observation Domain.

The random variable X_i represents the observation at time/step i that was emitted by the system. Although it can be discrete/continuous , we will handle discrete case now.

A HMM is specified as $\lambda = (T, E, \pi)$.

T is called - Transition Matrix, where $T(i, j)$ is the probability of switching from state i to state j.

$$T(i, j) = P(z_{k+1} = j | z_k = i)$$

E is called - Emission Matrix, where $E(i, k)$ is the probability of current state i and the observed emission is k.

$$E_i(x) = P(x | z_k = i)$$

π is the - starting probability, $\pi(i)$ is the probability that the system started in state i.

$$\pi(i) = P(z_1 = i)$$

The HMM satisfied the graphical model described the Trellis Diagram shown, thus we can write the joint probability as -

$$P(x_1, x_2, \cdots, x_n, z_1, z_2, \cdots z_n) = P(z_1) \cdot P(x_1|z_1) \prod_{k=2}^{n} P(z_k|z_{k-1}) \cdot P(x_k|z_k)$$

Using the parameters, we can rewrite the joint distribution as,

$$P(x_1, x_2, \cdots, x_n, z_1, z_2, \cdots z_n) = \pi(1) \cdot E_{z_1}(x_1) \prod_{k=2}^{n} T(z_{k-1}, z_k) \cdot E_{z_k}(x_k)$$

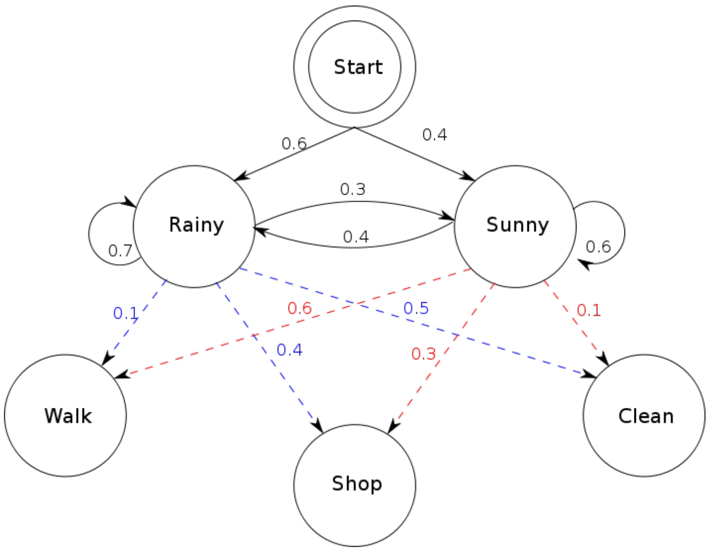

Figure 2: *A simple HMM*

The image above describes a simple HMM where the hidden states contain information about the day - rainy or sunny. The observed variables is whether a person walks, shops or cleans on any given day. These hidden states obviously impact the observations, rainy weather makes us stay indoors and clean. Sunny weather makes us go for a walk etc.

We might be able to observe the activities of a person and use these to infer the weather that day. Now there are different questions that we might want to ask. If a certain city has a specific weather pattern, what observations are now more likely ?. Given the observations, can we say something about the most likely weather patterns that caused it? etc.

Once we have the model and know its parameters, we can answers all sorts of questions by writing probability equations. We could even simulate a set of observations.

Although in this situation we are very clear on what the hidden states, this needn't be the case. Even with no prior knowledge of the system, we can make simple assumptions on number of states and see where it leads us.

One common toy example of HMM is feeding in raw texts of English articles, where each character is treated as an observation. Suppose it is used as raw data by someone who doesn't know the English language, who assumes that there are two hidden states and models it as a HMM. In this case, after running the Baum-Welch algorithm , our emission probabilities show a clear distinction between vowels and consonants. Thus, the HMM has learned an inherent structure in the English language.

3. 3 Questions

Question 1

Given the model $\lambda = (T, E, \pi)$ and a sequence of observations O, find $P(O|\lambda)$. That is, what is the probability that the sequence of observations was generated by the given model λ

Question 2

Given $\lambda = (T, E, \pi)$ and an observation sequence O, find the optimal state sequence given the observation sequence.

Question 3

Given an observation sequence O and the dimensions N and M, find the model $\lambda = (T, E, \pi)$ that maximizes the probability of O. Essentially, we estimate the parameter T and E such that, they maximise the likelihood of observing sequence O.

4. Algorithms

Here we discuss some algorithms and then we mention how to use them to answer the three questions. We assume the model λ is known. Let the observation sequence be $x = (x_1, x_2, \cdots, x_n)$

Forward-Backward Algorithm : Compute $P(z_k|x)$

Forward Algorithm : Compute $P(z_k, x_{1:k})\forall k$

Backward Algorithm : Compute $P(x_{k+1:n}|z_k)\forall k$

We note that

$$\begin{aligned} P(z_k|x) &\propto P(z_k, x) \\ &= P(x_{k+1:n}|z_k, x_{1:k}) \cdot P(z_k, x_{1:k}) \\ &= P(x_{k+1:n}|z_k) \cdot P(z_k, x_{1:k}) \end{aligned}$$

4.1. Forward Algorithm

We start of by expanding the probabilities.

$$\begin{aligned} P(z_k, x_{1:k}) &= \sum_{z_{k-1}=1}^{m} P(z_k, z_{k-1}, x_{1:k}) \\ &= \sum_{z_{k-1}=1}^{m} P(x_k|z_k, z_{k-1}, x_{1:k-1}) \cdot P(z_k|z_{k-1}, x_{1:k-1}) \\ &\qquad \cdot P(z_{k-1}|x_{1:k-1}) \cdot P(x_{1:k-1}) \end{aligned}$$

We can reduce the above equation using the markov properties into,

$$\alpha_k(z_k) = P(z_k, x_{1:k}) = \sum_{z_{k-1}=1}^{m} P(x_k|z_k) \cdot P(z_k|z_{k-1}) \cdot \alpha_{k-1}(z_{k-1})$$

Add to this $\alpha_1(z_1) = P(z_1, x_1) = P(z_1) \cdot P(x_1|z_1)$.

Thus we now have a recursive set of equations which can be computed starting from α_1

This is the Forward Algorithm and it can be used to answer question1.

$$P(x; \lambda) = \sum_{z=1}^{m} P(x, z; \lambda) = \sum_{z=1}^{m} \alpha_n(z)$$

4.2. Backward Algorithm

We start of by expanding the probabilities.

$$\beta_k(z_k) = P(x_{k+1:n}|z_k) = \sum_{z_{k-1}=1}^{m} P(x_{k+1:n}, z_{k+1}|z_k)$$
$$= \sum_{z_{k-1}=1}^{m} P(x_{k+2:n}|z_{k+1}, z_k, x_{k+1})$$
$$\cdot P(x_{k+1}|z_{k+1}, z_k) \cdot P(z_{k+1}|z_k)$$

We can reduce the above equation using the markov properties into,

$$\beta_k(z_k) = P(x_{k+1:n}|z_k) = \sum_{z_{k-1}=1}^{m} \alpha_{k+1}(z_{k+1}) \cdot P(x_{k+1}|z_{k+1}) \cdot P(z_{k+1}|z_k) \cdot$$

Add to this $\beta_n(z_n) = 1$.

Thus we now have a recursive set of equations which can be computed starting from β_n

This is the Backward Algorithm

Now, define $\gamma_k(i) = P(z_k = i|x)$, therefore

$$\gamma_k(z_k) = \frac{\alpha_k(z_k) \cdot \beta_k(z_k)}{P(x;\lambda)}$$

This can be calculated using the Forward and Backward Algorithm.

Now, $\max\limits_{z} \ \gamma_k(z)$ is the most likely state z at time/step k.

4.3. Viterbi Algorithm

Goal : Compute $z^* = \arg\max\limits_{z} P(z|x)$ where, $z = (z_1, z_2, \cdots, z_n)$ and $x = (x_1, x_2, \cdots, x_n)$

The viterbi algorithm is a Dynamic Programming Algorithm approach to finding the most likely state sequence.

First, note that for the two functions $f(a)$ and $g(a, b)$,

$$f(a) \geq 0 \ \ \forall \ \ a, \ \ g(a,b) \geq 0 \ \ \forall \ \ a, b$$
$$\Rightarrow \max_a \ \ f(a)g(a,b) = \max_a [f(a) \ \ \max_b \ \ g(a,b)]$$

Also, $\arg\max\limits_{z} P(z|x) = \arg\max\limits_{z} P(z,x)$

$$\mu_k(z_k) = \max_{z_{1:k}} \ \ P(z_{1:k}, x_{1:k})$$
$$= \max_{z_{1:k}} \ \ P(x_k|z_k) \cdot P(z_k|z_{k-1}) \cdot P(z_{1:k-1}, x_{1:k-1})$$
$$= \max_{z_{1:k}} \ \ [P(x_k|z_k) \cdot P(z_k|z_{k-1}) \cdot \max_{z_{1:k-1}} \ \ (z_{1:k-1}, x_{1:k-1})]$$
$$= \max_{z_{1:k}} \ \ [P(x_k|z_k) \cdot P(z_k|z_{k-1}) \mu_{k-1}(z_{k-1})]$$

By definition, $\mu_1(z_1) = \max\limits_{z_{1:1}} \ \ P(z_1, x_1) = P(z_1, x_1) = P(z_1) \cdot P(x_1|z_1)$

And, $\mu_n(z_n)$ gives us the required solution.

4.4. Baum-Welch Algorithm

This algorithm is used to estimate the parameters of the model given an observed sequence.

We define 'di-gammas' to be

$$\gamma_k(z_k, z_{k+1}) = P(z_k, z_{k+1}|x)$$

which is the probability of being in state z_k at time k and state z_{k+1} at time k+1.

We can see that,

$$\gamma_k(z_k, z_{k+1}) = \frac{\alpha_k(z_k) \cdot P(z_{k+1}|z_k) \cdot P(x_{k+1}|z_{k+1}) \cdot \beta_{k+1}(z_{k+1})}{P(x;\lambda}$$

And we have the relation,

$$\gamma_k(z_k) = \sum_{z=1}^{m} \gamma_k(z_k, z)$$

Given γ, Di-γ, the initial model $\lambda = (T, E, \pi)$, we can iteratively get a better estimate for the model as -

1. For $z = 1, \cdots, m$
$$\pi(z) = \gamma_1(z)$$
2. For $i, j \in 1, \cdots, m$
$$T(i,j) = \frac{\sum\limits_{k=1}^{n-1} \gamma_k(i,j)}{\sum\limits_{k=1}^{n-1} \gamma_k(i)}$$
3. For $j = 1, \cdots, m$ and $l = 1, \cdots n$
$$Ej(l) = \frac{\sum\limits_{\substack{k=1 \\ x_k=l}}^{n} \gamma_k(j)}{\sum\limits_{k=1}^{n} \gamma_k(j)}$$

The starting point for $\lambda = (T, E, \pi)$ should have values that are approximately described as -

$$\pi \approx \frac{1}{m}, \quad T(i,j) \approx \frac{1}{m}, \quad E_i(j) \approx \frac{1}{n}$$

,

So the steps of the BW Algorithm are -

1. Initialize $\lambda = (A, B, \pi)$
2. Compute the Forward,Backward,Gamma and Di-Gamma probabilities.
3. Compute the Iterative Estimates of the parameters.
4. If $P(x;\lambda)$ increases, repeat.

5. Carnatic Music

Carnatic Music is a style of Indian Classical Music that is prominent in the southern states - Tamil Nadu,Karnataka,Kerala and Andhra Pradesh. It is characterized by a melodic and and intricate combinations of swaras which is appealing and capable of capturing different emotions like anger,sadness,haste,joy etc.

The building blocks of a carnatic song are its Raga and Tala. A Raga is a collection of a subset of the musical notes. A Tala is similar to a beat. The song is composed of phrases of notes which obey the Raga and are overlayed onto a Tala. Raga as a form has its own dimensions and qualities. Every raga is a distinct musical entity by itself and possesses well defined characteristics. It consists of a series of notes which bear a definite relationship to adhara shadja and which occur in a particular sequence. The Arohanam is the ascending order of the raga, Avarohanam is the descending order of the raga.

The Raga system is considered as the foundation of Indian music. There are seven fundamental notes - S R G M P D N (with variations in each note). A parent Raga or a "Melakarta" Raga contains all the seven notes. A derived Raga or a "Janya" Raga contains a subset of the notes of a particular Melakarta Raga.

The variations in the fundamental notes results in different Melakarta Ragas. The circular chart shows all the 72 possible combinations. Each of these 72 Melakarta ragas have numerous Janya Ragas. The scheme of 72 ragas is mathematical and thus gives the Raga system a definitive scientific structure.

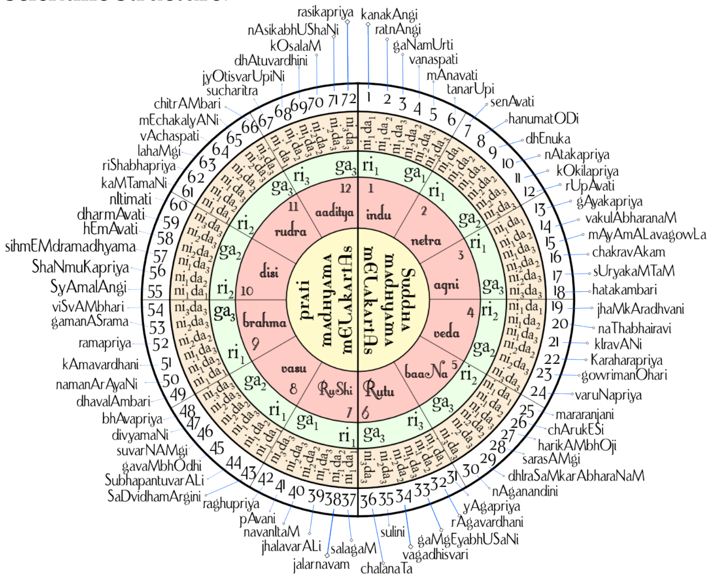

Figure 3: *The Melakartha Raga Chart.*

6. Classification with HMMs

The structure inherently contained in the Raga leads us to the expectation that there can be probabilistic models where we treat each song as a temporal sequence of notes.

I would like to think that the greatest composers in carnatic music had trained their brains to recognize patterns that occur often and that are pleasing to hear and inherently had a complex system of probabilistic models whose parameters were estimated with every real life practice session and eventually were so fine tuned that they could sample and generate musical sequences of arbitrary lengths from their inherent models for each raga and each tala. But of course, that could just be my scientific curiosity to mathematically capture everything around us. Nevertheless, I thought it was worth a try to see what the results would be.

Here we consider the problem of identifying the raga from a finite short musical sequence. In real-life the listener in a concert are constantly doing the sound processing task and trying to identify the ragas.

As an avid but amateur listener, I find it frustrating to be able to determine. The determination frequently gets narrowed down to a few suspect Ragas who are all similar and further distinction can be a lot harder.

Here I have chosen one particular Melakartha Raga - Dheera Shankabharanam (or just Shankarabharanam). I have only chosen one Melakartha Raga as, usually its easier to differentiate and assert about the melakartha raga as compared to distinguishing between janya ragas.

This is because, the Melakartha Ragas differ in their fundamental notes giving them a different sound.

I have tried to identify whether a particular sequence is the Melakartha Raga - Shankarabharanam or its derivatives - Arabhi, Bilahari, Hamsadhwani.

The details about all the Ragas considered is given in Table 1.

Note that a particular sequence can look like Bilahari but actually be from the Parent Raga Shankabharanam itself. This might happen because wihtin that particular sequence the characteristics of Shankarabharanam might not have been fully expressed. This gives further weight to probabilistic modelling as compared to plain statistical models or brute force checking. For eg. the sequence "S G P D" can belong to either of Shankarabharanam or Bilahari.

6.1. Data

The biggest obstacle was obtaining clean data to work with. Most of Carnatic Music is still learnt in the old fashioned way with handwritten notes and taught vocally rather than reotes.

The sound processing approach was not taken as this involves a much tougher problem of identifying the musical tone, pitch etc from raw files that are subject to the particular singer's coherence in voice and delivery and recording quality.

We are mostly concerned with the information containing within the notation and the songs in this project. There are few websites which have notes of some songs in the electron formats. Sheet music is not as popular in Carnatic Music as with Western or other forms.

Based on my personal training, I chose about 10-12 songs that I already knew (so that I could verify the notation given on the website as they didn't seem to be standardized.) After obtaining the raw notation data, there was some minimal data-cleansing to be done to convert this into a string of the symbols used in the song.

6.2. Training and Testing

Once the song sequence is extracted, we can train Hidden Markov Models on it. Four Hidden Markov Models have been one for each Raga - Shankarabharanam , Arabhi, Bilahari, Hamsadhwani.

After the training is complete, we sampled short sequences of a fixed length repeatedly and calculated the forward probabilities and thus the probability of the sequence wrt each HMM.

The confusion matrices give us a picture of the accuracy and precision of each HMM in capturing the particular Raga.

Short sequences were also sampled from two Ragas different from the ones we trained on - Ataana, Shudda Saveri and we tried to classify it to be one of the training Ragas. This gave us an insight into which training Raga most closely resembled it.

All the results have been shown in the tables and figures. In the histograms related to Hamsadhwani we can easily see that the 3 states each correspond to a different set of swaras. State B for example seems to exclusively correspond to the higher swaras.

We can also see from the confusion matrix for Hamsadhwani that it is highly successful in classifying a sequence to be from Hamsadhwani Raga. Where-as, the Shankarabharanam model seems much less successful. This can be due to the fact that a shankarabharanam sequence can easily look very similar to that of Bilahari etc. These results also depend on the length of the sequences tested and the source of these sequences. Experimentation shows that a length of 10 is usually not enough and that a length of 40 or above can lead to inaccurate probabilities (as the probability values get very low). A length of 20 was chosen to create the tables.

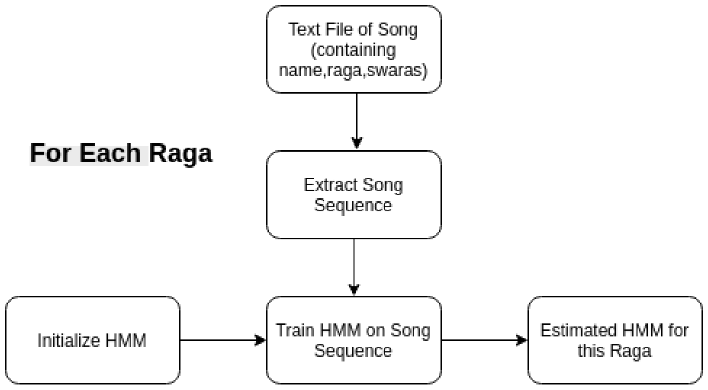
For Each Raga
Text File of Song (containing name,raga,swaras)
Extract Song Sequence
Initialize HMM
Train HMM on Song Sequence
Estimated HMM for this Raga

Figure 4: *Training Work flow*

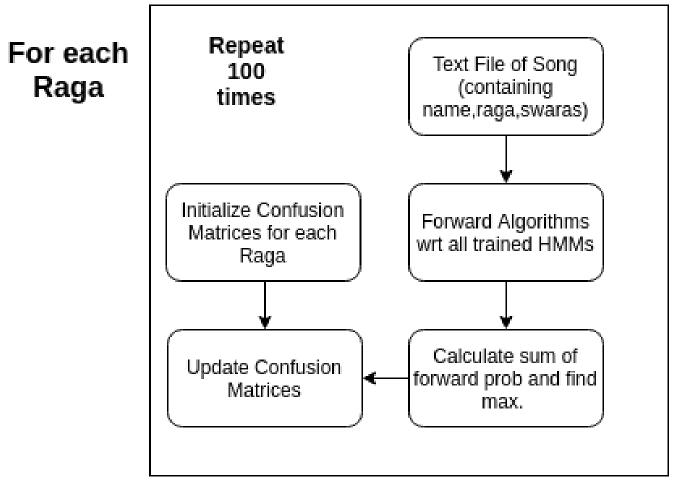
For each Raga
Repeat 100 times
Text File of Song (containing name,raga,swaras)
Initialize Confusion Matrices for each Raga
Forward Algorithms wrt all trained HMMs
Update Confusion Matrices
Calculate sum of forward prob and find max.

Figure 5: *Testing Work flow*

Table 1: *Ragas chosen and their properties*

	Aarohanam	Avrohanam
Shankara.	S R2 G3 M1 P D2 N3 S	S N3 D2 P M1 G3 R2 S
Arabhi	S R2 M1 P D2 S	S N3 D2 P M1 G3 R2 S
Bilahari	S R2 G3 P D2 S	S N3 D2 P M1 G3 R2 S
Hamsadh.	S R2 G3 P N3 S	S N3 P G3 R2 S
Ataana	S R2 M1 P N3 S	S N3 D2 P M1 P G3 R2 S
S.Saveri	S R2 M1 P D2 S	S D2 P M1 R2 S

7. Results

We can look at the trained HMMS

Table 2: *Final Transition Matrix of the estimated HMM Model - Shankarabharanam*

	A	B	C
A	0.07	0.33	0.59
B	0.17	0.77	0.06
C	0.79	0.02	0.19

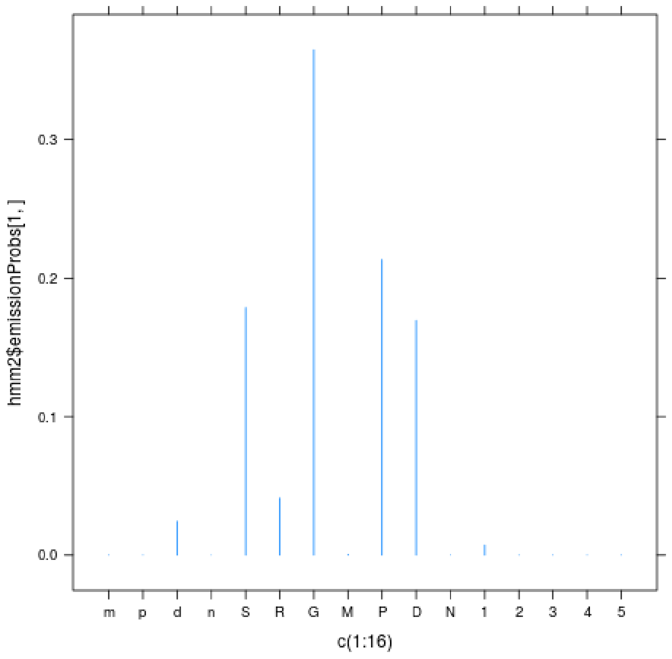

Figure 6: *Bilahari State A Emission*

Table 3: *Log of Final Emission Matrix of the estimated HMM Model - Shankarabharanam*

	A	B	C
m	-Inf	-Inf	-Inf
p	-19.981003	-4.097038	-45.092754
d	-23.612328	-3.809356	-53.407804
n	-6.832919	-2.495907	-3.919633
S	-1.666490	-6.987564	-2.487120
R	-3.504350	-7.195796	-1.233545
G	-1.380197	-2.624693	-12.413387
M	-11.365407	-1.304033	-3.222295
P	-2.922914	-1.012430	-8.254159
D	-4.961158	-1.772820	-2.358310
N	-1.349444	-10.241079	-6.592482
1	-2.627366	-17.691503	-1.016071
2	-2.200455	-29.821646	-3.572172
3	-22.275888	-30.192464	-2.531284
4	-3.680154	-61.639964	-31.917195
5	-Inf	-Inf	-Inf

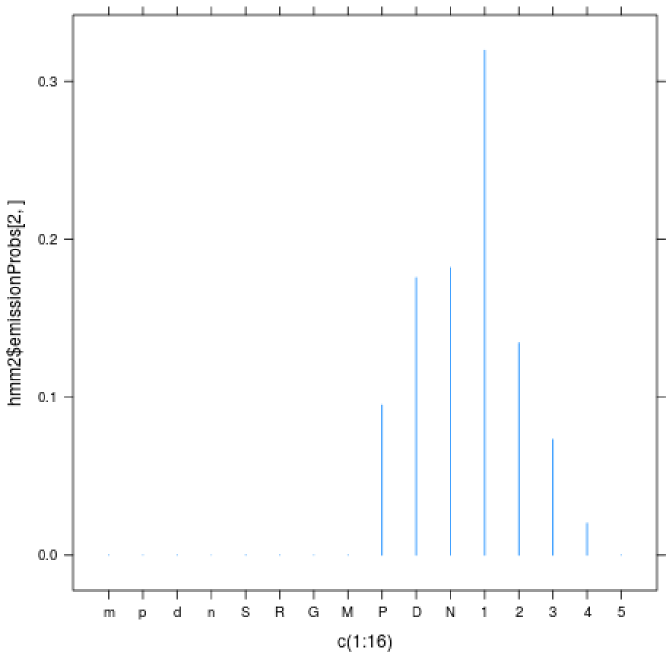

Figure 7: *Bilahari State B Emission*

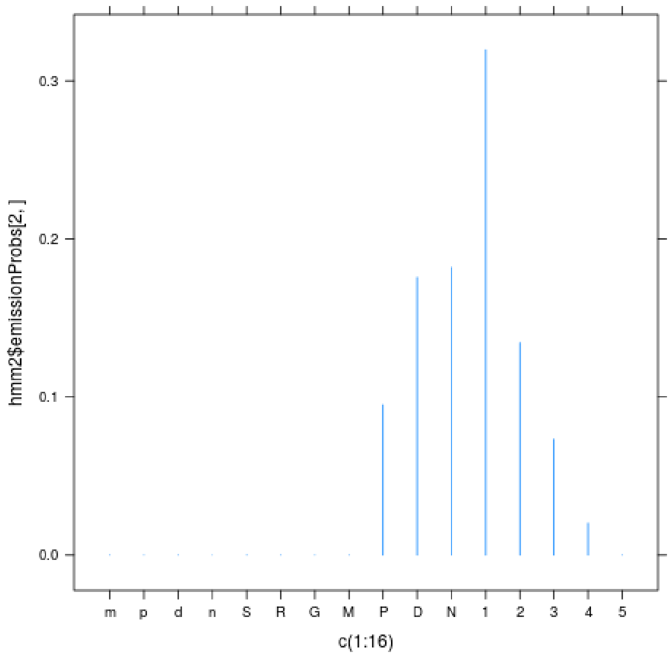

Figure 8: *Bilahari State C Emission*

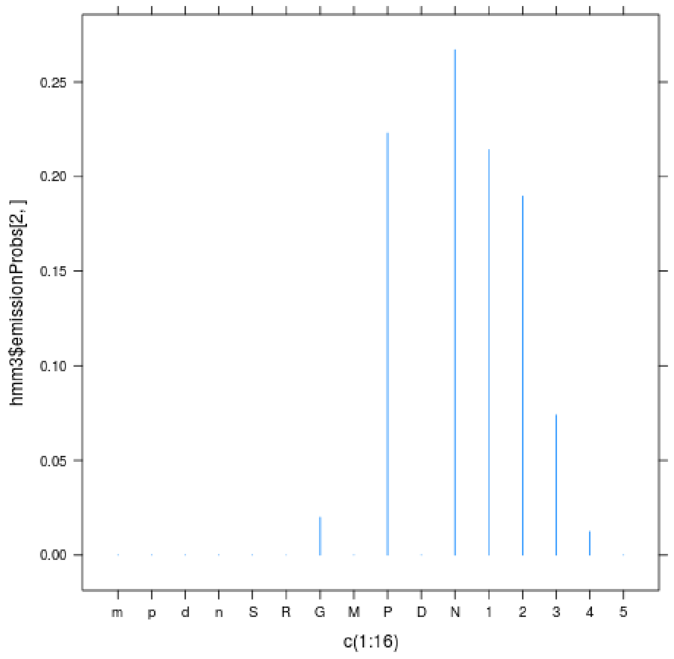

Figure 10: *Hamsadhwani State B Emission*

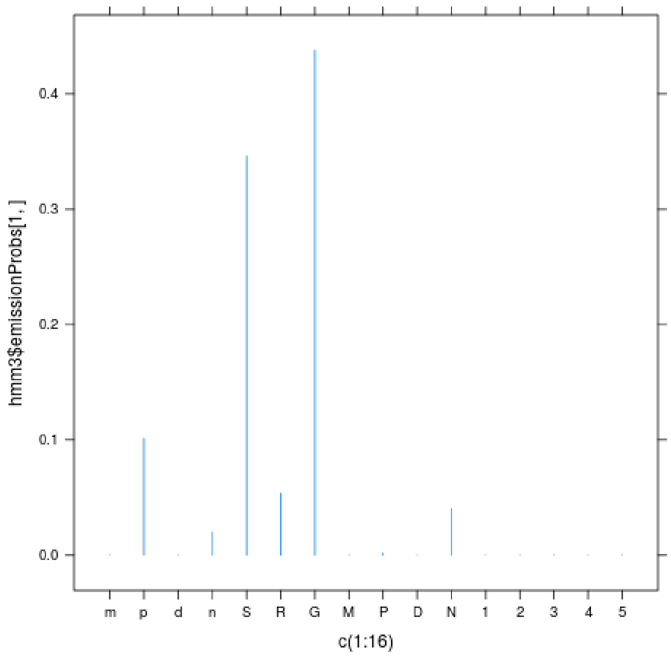

Figure 9: *Hamsadhwani State A Emission*

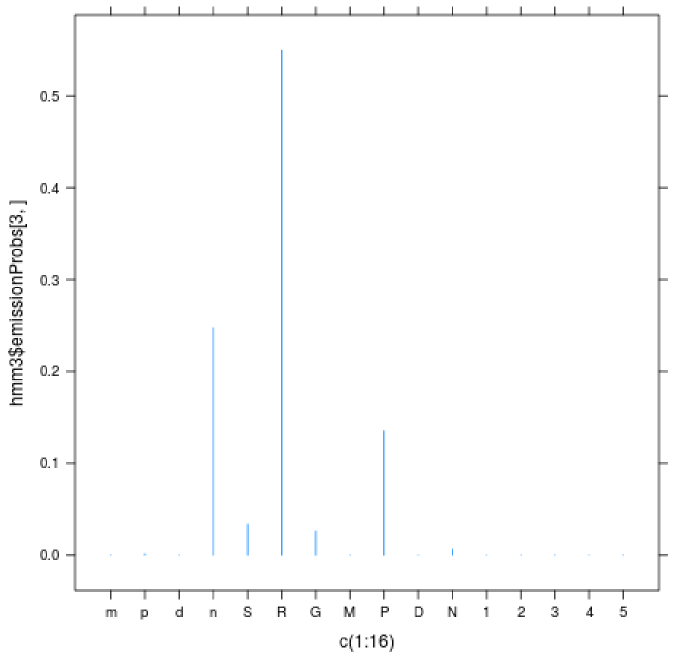

Figure 11: *Hamsadhwani State C Emission*

Table 4: *Confusion Matrix for HMM Model - Shankarabharanam*

	Predicted Positive	Predicted Negative
Actual Positive	42	58
Actual Negative	63	237

Table 5: *Confusion Matrix for HMM Model - Arabhi*

	Predicted Positive	Predicted Negative
Actual Positive	79	21
Actual Negative	0	300

Table 6: *Confusion Matrix for HMM Model - Bilahari*

	Predicted Positive	Predicted Negative
Actual Positive	62	38
Actual Negative	58	242

Table 7: *Confusion Matrix for HMM Model - Hamsadhwani*

	Predicted Positive	Predicted Negative
Actual Positive	96	4
Actual Negative	0	300

Table 8: *Performance Measure of HMMs*

	Shankara.	Arabhi	Bilahari	Hamsadh.
Accuracy	0.70	0.95	0.76	0.99
Precision	0.40	1.00	0.52	1.00
Recall/TP	0.42	0.79	0.62	0.96
F-measure	0.08	0.19	0.11	0.20

Table 9: *Confusion Matrix for Other Test Data*

	Shankara.	Arabhi	Bilahari	Hamsadh.
Ataana	-231.54	-Inf	-237.53	-Inf
Shudda Saveri	-242.65	-271.36	-284.80	-Inf

8. Automatic Music Generation with HMMs

Once we have trained we can simulate each of these HMMs to generate musical sequences for arbitray lengths.

One such example resulted in the following sequence from the Hamsadhwani HMM

```
states
"A" "A" "C" "C" "A" "A" "B" "B" "B" "B" "B" "B" "B"

observation
"G" "G" "R" "R" "S" "G" "3" "N" "1" "N" "4" "2" "1"
```

9. Future Work and Conclusions

There are still much more that can be done. Some directions of further progress could be -

1. More Data. Preparing a larger dataset.
2. More Ragas. Train HMMs on more Ragas.
3. Account for Tala and "Swara-pauses"
4. Use other models like the Trigramm HMM that predict based on current state and the previous state.

In conclusion, HMMs are a start and they predictably work well enough. The sequences simulated look pretty decent but sometimes we can observe drastic changes in successive observations. This calls for some amount of "smoothing", the Trigram HMM could help resolve this. Although some of the accuracy looks low, we have certainly taken those Ragas that are known to be similar and then applied. Also, the HMM of a Raga has been trained on one song of that Raga. There are many Ragas that can be expressed very differently in different songs , for eg. the same Raga in songs that are meant to be joyous can look different from another song meant to be mellow. Also there can be variations among particular composers and styles. Its possible that our HMM inadvertently captured information about the composer and differentiated a test sequence to be negative although its the same Raga as its not the same style as the composer of the training song. Such variations can be resolved largely by taking multiple sample songs and having an ensemble of HMMs.