Speech Recognition using Hidden Markov Models

Vlad Slyusar, Stephen Giardinelli

ECE 5831: Pattern Recognition & Neural Networks
University of Michigan

December 12, 2016

0.1 Problem Statement

To most humans, speech recognition feels rather intuitive as the deep neural processing required to support the task generally takes place subconsciously. Nonetheless, speech recognition is a highly complex process that humans develop mostly through years of cognitive training during childhood. The ability for a machine to recognize such patterns, with at least similar precision, can provide a vast multitude of useful applications ranging from automating stenography within a courtroom to helping a child with hearing disabilities. Unfortunately, the same process that comes naturally to humans provides serious challenges in implementation on a computer. Historically, both deterministic and statistical model approaches have been developed around the world and demonstrated to various levels of success. While a deterministic model generally exploits some specific properties of a signal, the statistical model approach attempts to characterize the signal only by its statistical properties[Rab89]. The focus of this paper centers on developing a statistics-based approach to voice recognition using hidden Markov processes.

0.2 Current State of the Art

As in many specializations within the field of machine learning, neural networks seem to be at the state of the art for speech recognition. According to Saon et al, combining different network architectures allowed for a record low of 6.6% error rate on the Hub5 2000 dataset. This was achieved by combining a recurrent neural network and a deep convolutional neural network. Currently, Saon et al.'s model achieves within 3% of average human performance, which is a very impressive result[SSRK16].

0.3 Method

Before hidden Markov models can be implemented, the theory behind the process must be understood. Lets start with a statistical Markov model in which the system being modeled is assumed to be a Markov chain with observable states and their respective transition probabilities, as represented by the following elements:

$$\begin{gathered}\text{States: } S = S0, S1, ...Sn \\ \text{Transition Probabilities: } P(q_t = S_i | q_{t-1} = S_j) = a_{ji}\end{gathered} \tag{1}$$

The Markovian assumption states that transition probability depends only on the current state. Thus the probability can be represented as:

$$\begin{aligned}P(q_t = S_i | q_{t-1} = S_j, q_{t-2} = S_k) = P(q_t = S_i | q_{t-1} = S_j) = a_{ji} \\ a_{ij} \geq 0 \forall j, i, \ \sum_{i=0}^{N} a_{ji} = 1\end{aligned} \tag{2}$$

While the Markov model provides a simple and elegant solution, its application is only useful when the states can be observed. In the case of a Markov chain with non-observable states, the hidden Markov model approach must be taken. While the outputs at each state are not directly observable, the probability distribution of the outputs at each state can be computed. Lets represent the output probability distribution, at state j and for symbol k, as:

$$P(y_t = O_k | q_t = S_j) = B_j(k) \tag{3}$$

The output that can be observed from the hidden state follows a multivariate Gaussian distribution with a mean vector μ and covariance matrix Σ for each of the states. Let the matrix A represent the transition probability from state i to state j. The probability for the chain to start in state i can be represented by π_i. The model parameters for the HMM can be represented by λ:

$$\lambda = \{A, \pi, \mu, \Sigma\} \tag{4}$$

We then use a forward-backward algorithm to compute the probability:

$$P(O|Q, \lambda) = \prod_{t=1}^{T} P(O_t | q_t, \lambda) \tag{5}$$

where observation sequence:

$$O = O_1, O_2, ..., O_T \tag{6}$$

and state sequence:

$$Q = q_1, q_2, ..., q_T \tag{7}$$

Because of the computationally intensive nature of (5) the forward-backward algorithm provides us a way to compute this probability in a more efficient manner [Bis06]. For this algorithm, we define a forward variable:

$$\alpha_t(i) = P(O_1 O_2 ... O_t, q_t = S_i | \lambda) \tag{8}$$

We solve for $\alpha(i)$ like so:

1. Initialization

$$\alpha_1(i) = \pi_i b_i(O_1) \quad 1 <= i <= N \tag{9}$$

2. Induction

$$\alpha_{t+1}(j) = [\sum_{j=1}^{N} \alpha_t(j) a_{ij}] b_j(O_{t+1}) \quad \begin{array}{l} 1 \leq t \leq T-1 \\ \\ 1 \leq j \leq N \end{array} \tag{10}$$

3. Termination

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{11}$$

Furthermore, we have the backward algorithm to solve for our backward variable:

$$\beta_t(i) = P(O_{t+1} O_{t+2} ... O_T | q_t = S_i, \lambda) \tag{12}$$

1. Initialization

$$\beta_T(i) = 1, \quad 1 \le i \le N \tag{13}$$

2. Induction

$$\beta_t(j) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \quad 1t = T-1, T-2, ...1 \quad 1 \le i \le N \tag{14}$$

In order to train the model, we used an expectation-maximization algorithm - specifically the Baum-Welch algorithm [DLR77]. For the sake of brevity we will leave out the details of this algorithm as they can be acquired from the original paper. We use the Baum-Welch algorithm as an iterative approach to update the values of π, A, μ, and Σ.

0.4 Implementation Details

We based our implementation on a similar project developed by Hakon Sandsmark [San10] and our approach consisted of four main steps. First we imported our training data and performed some feature extraction, we then trained our HMM's using the Baum-Welch alorithm. Finally we imported our test data, extracted the features as we did for the training data, and evaluated the probability that the observations were produced by each HMM (trained for each word). We then classify the input, based on the HMM with the highest probability.

All of our data, both test and training, was recorded using Audacity in quiet but uncontrolled environments. The recording was done on a single (mono) channel at 44.1 kHz. Once in MATLAB, the all audio files were down-sampled to 8 kHz. We broke the signal up into blocks of 40 samples with an overlap of 20 samples. We then multiplied each block of samples by a Hamming window, and finally performed a fast Fourier transform to represent the signal in the frequency domain. Our features consisted of the F largest maxima from the frequency spectrum in each sample window. Thus our features consisted of an array of M vectors each consisting of F values pertaining to the 6 highest peaks in the frequency domain signal. M is a value that will differ for each audio file based on the length of the clip, and F is a tunable parameter that we experimented with to find optimal results.

Our model consisted of five Hidden Markov Models (HMMs) - one for each word. Each HMM consisted of N states, where N was an important model parameter. We then trained each model on ten samples of the same word. The flow chart of the system is shown in figure 1.

In order to train each HMM, we used the Baum-Welch [DLR77] algorithm - an iterative expectation maximization algorithm as previously described. The output generates a hidden Markov model for each of the training words. The parameters λ are continuously updated until a satisfactory model is achieved.

$$\lambda = \{A, \pi, \mu, \Sigma\} \tag{15}$$

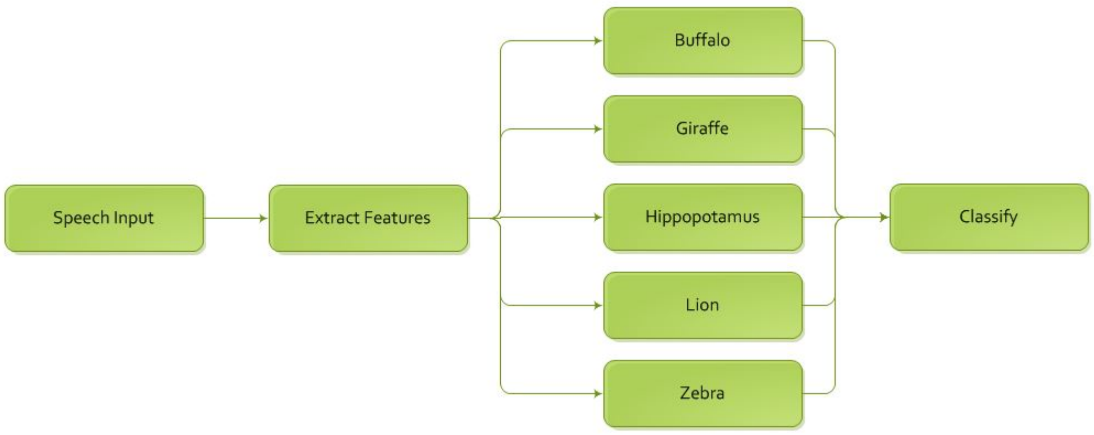


Figure 1: High level flow chart of the system

Finally, we used a forward-backward algorithm to compute the probability:

$$P(O|Q,\lambda) = \prod_{t=1}^{T} P(O_t|q_t,\lambda) \tag{16}$$

This probability can tell us the probability that our observations came from a given HMM. So this equation is evaluated for each HMM to find the HMM that would most likely produce the given observations. Our classifier works by classifying the input as the word pertaining to the HMM with the highest probability of (16).

0.5 Dataset Used

A custom dataset was created for the project implementation consisting of five different words. The following phrases were selected:

- Buffalo
- Giraffe
- Hippopotamus
- Lion
- Zebra

Each of the words was recorded 10 different times and separated into individual samples for training the HMM. Additional samples were recorded for testing the recognition capability of the trained system on novelty data. Furthermore, additional samples of each word were recorded by a second speaker with the intent of testing against the HMMs generated solely from the first speaker samples. Figures 2 and 3 illustrate the differences in voice across two separate speakers.

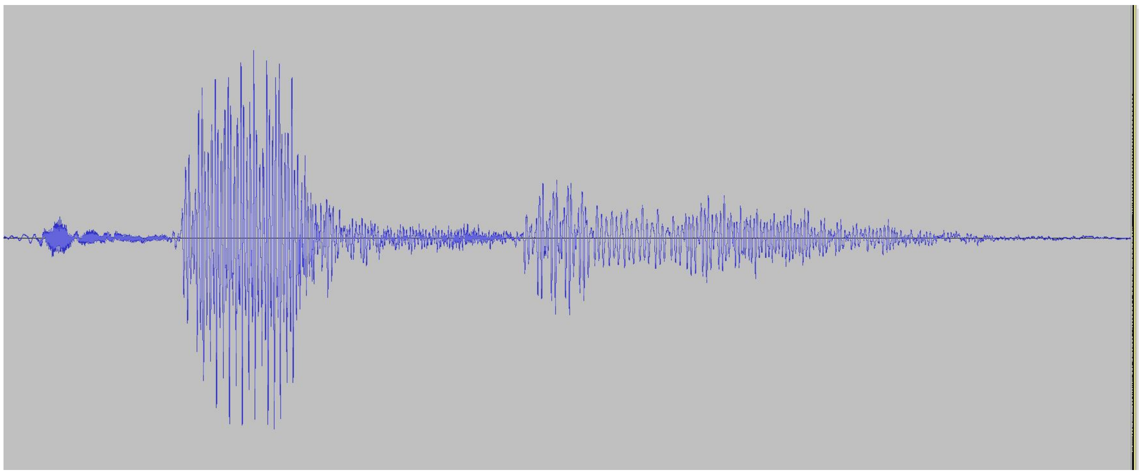


Figure 2: Sound sample - "buffalo" - speaker 1

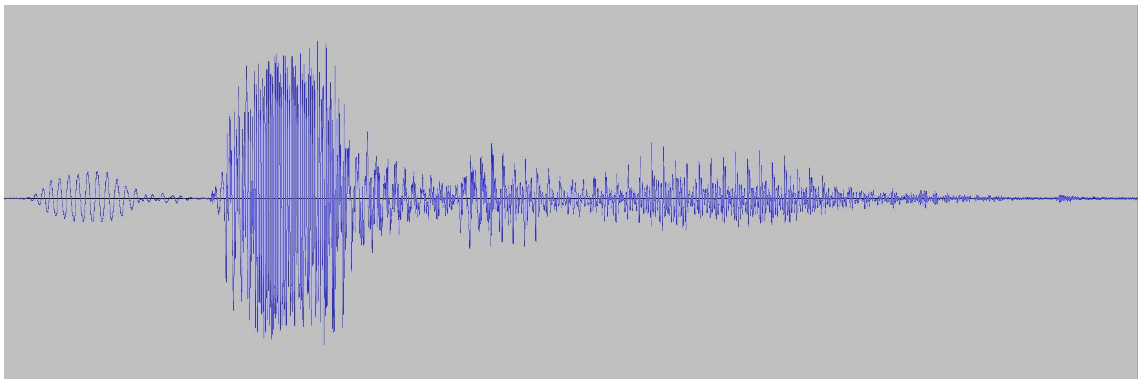
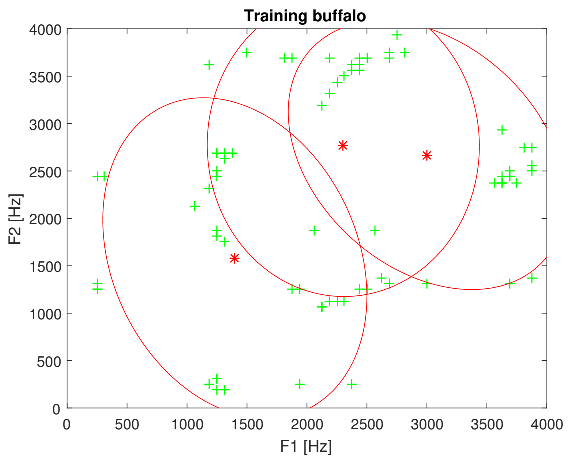


Figure 3: Sound sample - "buffalo" - speaker 2

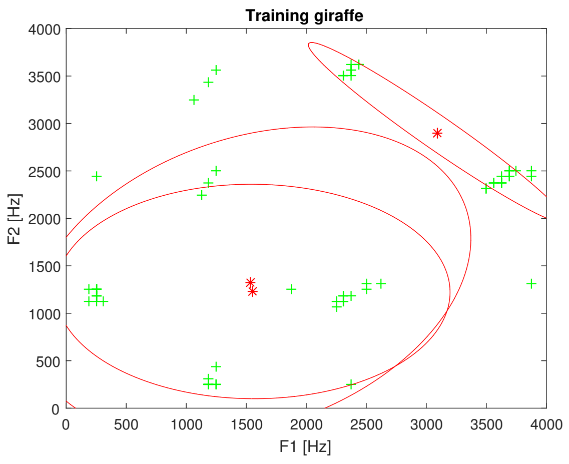
0.6 Results

Applying the Baum-Welch algorithm to the five training samples from speaker one, each with ten iterations, produced the Gaussian plots provided below in 4. The model has three states, each with their own Gaussian shown on a two-dimensional plot spanning the two most prominent frequencies F1 and F2. The points, shown as green '+' characters, represent individual frames from one of the training samples while the red '*' points show the mean values of each Gaussian. Each ellipse around the mean encloses the area corresponding to a 3/4 confidence interval.

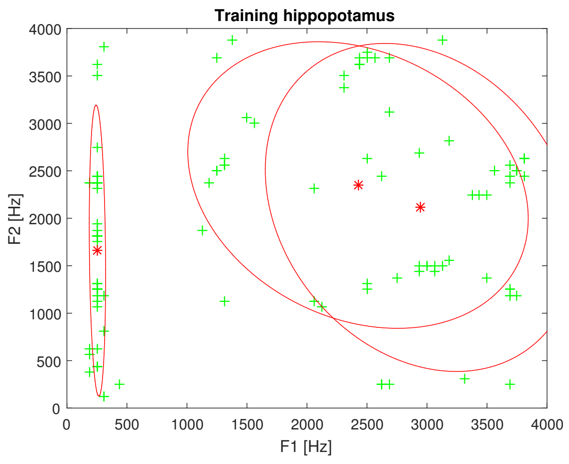
Once each of the HMMs was trained, a new set of samples from speaker 1 were tested. Using the 5 hidden Markov models for the individual words, an additional word prevously unseen by the model was processed for classification. A sample output is provided in table 1, this data trained on 10 iterations of each word and tested an 11th recording againts all the HMMs to determine closest match and classify. Additionally, the test was performed on speech samples from speaker 2. In this case, the HMMs trained by samples from speaker 1 were compared with the new recording for classification. The results are provided in table 2.



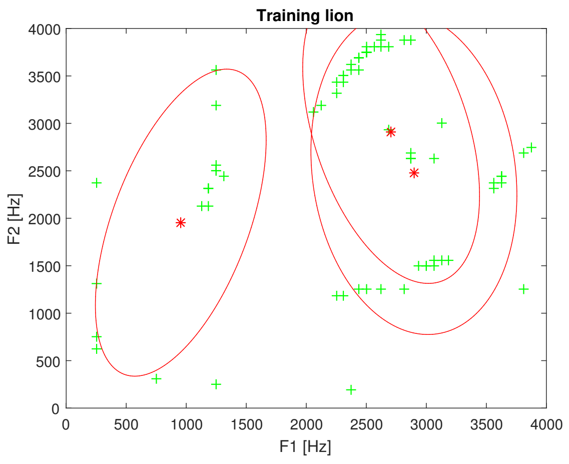
(a) Training Buffalo



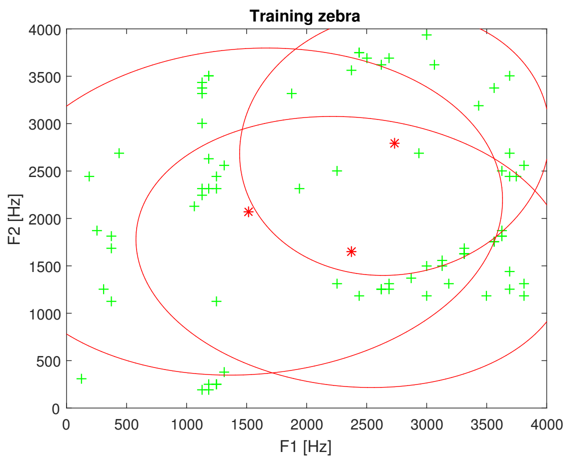
(b) Training Giraffe



(c) Training Hippopotamus



(d) Training Lion



(e) Training Zebra

Figure 4: Fitted Gaussian plots for training samples from speaker 1.

Classified as:	Classified as:	Classified as:	Classified as:	Classified as:
Buffalo	Giraffe	Hippopotamus	Hippopotamus	Zebra
Correct Answer is:	Correct Answer is:	Correct Answer is:	Correct Answer is:	Correct Answer is:
'Buffalo'	'Giraffe'	'Hippopotamus'	'Lion'	'Zebra'

Table 1: Sample Classification on Novel Speech Sample - Same Speaker

Classified as:	Classified as:	Classified as:	Classified as:	Classified as:
Hippopotamus	Giraffe	Hippopotamus	Hippopotamus	Zebra
Correct Answer is:	Correct Answer is:	Correct Answer is:	Correct Answer is:	Correct Answer is:
'Buffalo'	'Giraffe'	'Hippopotamus'	'Lion'	'Zebra'

Table 2: Sample Classification on Novel Speech Sample - New Speaker

0.7 Discussion on the results

The recognition was successful on 4/5 samples, in this case miss-classifying 'Lion' as 'Hippopotamus' as can be seen in table 1. In fact, over numerous trials the majority of errors encountered involved the word "hippopotamus". A potential reason for such behavior lies within the phoneme count of the word, "hippopotamus" being significantly longer, and for a small number of hidden states leads to a loss of some temporal information. Over 10 trials, an average of 3.6/5 of the words were properly recognized providing an effective recognition rate of 72%.

When performing an equivalent test but with speech samples from a new speaker, the system performance decreased significantly. In the provided test case presented in table 2, only 3/5 samples were recognized, providing an average of 2.8/5 correct classifications over 10 consecutive trials. Keeping the sample content and length relatively similar, a different speaker source achieved an effective recognition rate of 56%.

The best results were had when the audio input was down-sampled from 44.1kHz to 8kHz, the audio sample was broken up into blocks of 40 samples (fs=40) with an overlap of 20 samples (overlap =20), six frequency maxima were located per sample block (D=6), and three states were used in each HMM (N=3). These aforementioned parameters were experimented with to find an optimal solution. Down sampling turned out to be extremely important, as a 44.1kHz signal had far much much high frequency noise. Down sampling to 8kHz or even 4kHz yielded much more accurate results.

The sample block size and overlap were found to yield accurate results at the above listed values, however these values should be scaled with the sampling frequency of the input. Finally we ended up choosing 3 states per HMM as our experimentation yielded the best results with this value.

0.8 Conclusion and Future Directions

Working with Hidden Markov Models proved to be a difficult and convoluted process, however the results from HMM's seem to be acceptable for the level of complexity of the software. Based on our research neural networks, both convolutional and recurrent, have far superior accuracy especially when combined. However, the level of complexity of designing a convolutional or recurrent neural network is far greater than our modest HMM model. In our future research we would like to experiment with neural networks and compare the accuracy and performance of neural networks to HMM models.

Bibliography

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[DLR77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[Rab89] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989.

[San10] Hakon Sandsmark. Isolated-word speech recognition using hidden markov models. Dec 2010.

[SSRK16] George Saon, Tom Sercu, Steven J. Rennie, and Hong-Kwang Jeff Kuo. The IBM 2016 english conversational telephone speech recognition system. *CoRR*, abs/1604.08242, 2016.