Bird Speech Recognition

Apurv Varshney, Gaurav Ganna and Pratyay Gaikwad
Team 5

November 25, 2018

1 Introduction and Overview

Classification of birds by their sound has many potential applications in ecological surveillance, conservation monitoring, taxonomy, and vocal communication studies. Using audio recordings rather than pictures is justifiable since bird calls and songs have proven to be easier to collect and to discriminate better between species. But for it to be useful for the layman, it should be scalable, should work in noisy environments and on different lengths.

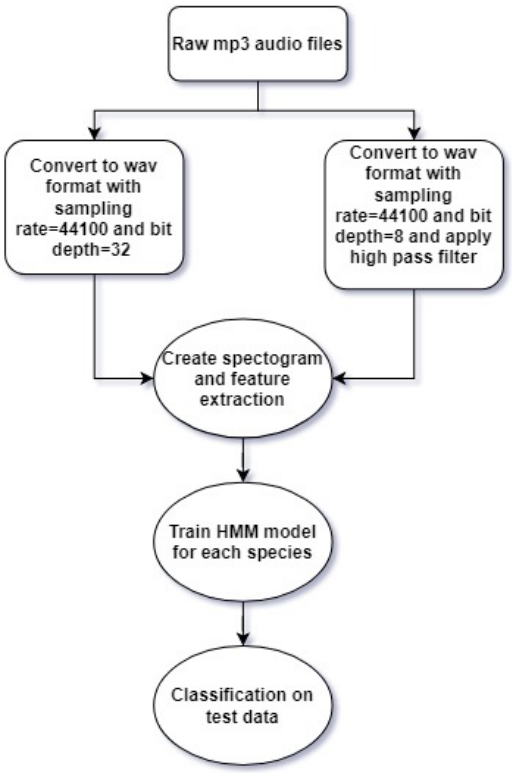

Figure 1: Flow-Chart

Related work

This experiment [2] uses machine learning to organize thousands of bird sounds and cluster similar sounds using a technique called t-SNE.

This paper [5] classifies bird sounds using unsupervised feature learning. This method is unique in the sense that it does not require any training labels or any other side-information. This method shows promising accuracy but is computationally intensive.

2 Methods

2.1 Spectogram Creation

As a common technology to visualize the result of a short-time Fourier Transform(STFT), the spectrogram is a stack of multiple spectrums from a time series. The x-axis is usually time and the y-axis is usually frequency. When we intercept a vertical frame from a spectrogram, a column of amplitude values will be generated, which is usually represented by a colormap in a spectrogram figure. The process of spectrogram generation is performed in a conventional way: (a) First, cut the signal data into several segments with a fixed length(usually an integer power of 2 for Fast Fourier Transform(FFT) efficiency). The segment length in our study is 4096(212). (b) Apply a window on the segment by taking convolution of the segment and a windowing function. The window function applied is Hamming Window. (c) Take FFT of the segment, thus gaining a spectrum of the segment. (d) Arrange all spectrum together in time order.

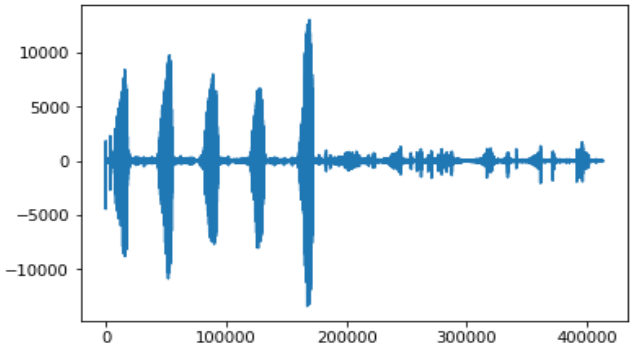

Figure 2: Simple wave plot of a Bird specie

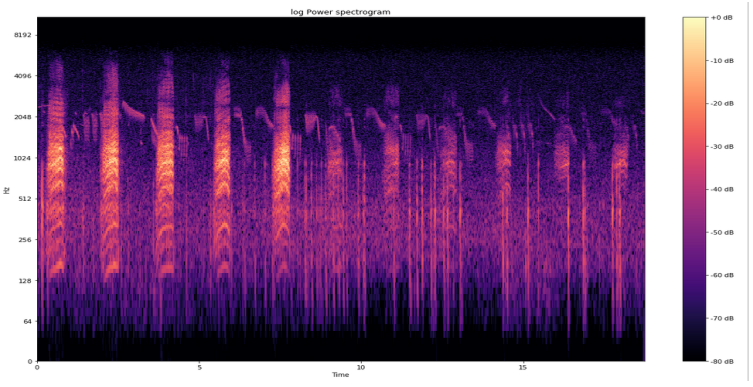

Figure 3: Spectogram of a bird specie

2.2 Feature Extraction

Our feature extraction strategy is to take the maximum amplitude $A_{max,t}$ and the corresponding frequency $f_{max,t}$ in each time frame t in the spectrogram. This will roughly describe the shape of a specific pattern in a spectrogram. The feature vector is hence

$$O_t = (A_{max,t}, f_{max,t}).$$

We have kept 80% of our dataset for training and rest is for testing our model.

2.3 Gaussian Hidden Markov Model(HMM)

After segmentation, we get a sequence of vectors $O = O_1,..., O_t$ for each data sample. We first group training data into subsets by bird type Table 1. Then an HMM is fitted to each subset. As a result, an HMM network is obtained in which each node is a model with respect to a specific bird type. For the features extracted, we assume their probability distribution follows the Gaussian normal distribution. We also test our model with numbers of hidden states between 2 to 20 and compare respective results in order to find the optimal number of hidden state. Now with our HMM model fitted on the training set, we put testing dataset into the network for prediction. Now the model calculates the log likelihood for a testing dataset to happen under each of HMM nodes, and output the predicted bird type with the maximum log likelihood.

3 Experimental Analyses

Datasets

We initially collected recordings of birds found in Goa as one dataset(Dataset A). We then collected quality 'A' recordings of 7 species of variable length from xeno-canto website[1] as the second dataset(Dataset B). These recordings contain flight calls along with bird-songs. These were recorded in natural settings so it also includes long pauses, environmental noises along with actual bird audio.

Bird's Name	No. of Recordings
Mallard	8
Black hawk-eagle	7
Red-winged blackbird	7
Red-legged seriema	8
Sedge wren	8
Common whitethroat	8
Mourning dove	8

Table 1: Dataset B

Results

On testing our model on dataset A, we got less than 50% accuracy. Upon looking further we found that this was caused by the fewer number of recordings and bad quality recordings

Upon testing our model on various number of hidden states for dataset B, a respectable 78% accuracy was achieved.

Our project shows that it is not always needed to use too many dimensions of features as observance sequence in order to obtain a better fitting result. Instead, we only need two features, the maximal magnitude, and the corresponding frequency. We also inferred that more hidden states turn out to be useless in improving the results after a fixed number of hidden states. This is controversial to our previous knowledge of data fitting and a curse of dimensionality [3] is believed to happen.

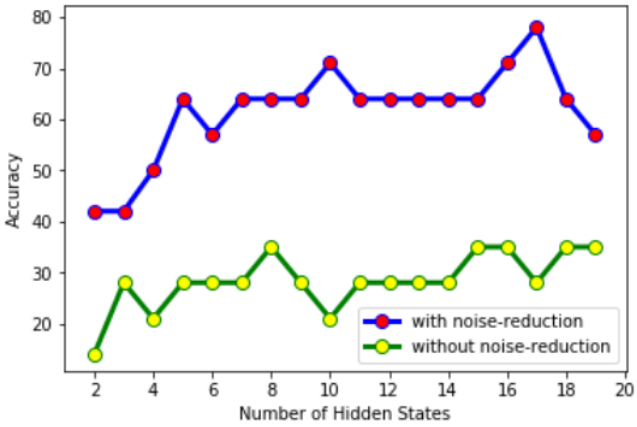

Figure 4: Accuracy vs Number of hidden-states

Our project gave convincing results that Hidden Markov Models can be applied to audio files of different lengths. One more thing that we concluded is noise reduction as a pre-processing step significantly improves the accuracy.

4 Discussion and Future Directions

The pre-processing step of our model is very slow and can be improved by various other methods like Mel Frequency Cepstral Coefficient [4], Mute Area Skipping, Root Mean Square (RMS) energy normalization for each spectrogram and median-based thresholding as done in this paper [5].

We can use advanced classifiers other than HMM to significantly improve results. We can also extend our model to classify multiple bird species in a recording.

This project can be used to create a user-friendly web application which can be accessed by anyone without any expert knowledge.

References

[1] https://www.xeno-canto.org/

[2] https://experiments.withgoogle.com/ai/bird-sounds/view/

[3] https://en.wikipedia.org/wiki/Curseofdimensionality

[4] http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

[5] https://peerj.com/articles/488/

[6] https://waterprogramming.wordpress.com/2018/07/03/fitting-hidden-markov-models-part-ii-sample-python-script/amp/