Chord Recognition and Progression Predictor

Talia Berdichevsky
berdichevsky.t@northeastern.edu
Northeastern University
Boston, Massachusetts, USA

Gianna Mattessich
mattessich.g@northeastern.edu
Northeastern University
Boston, Massachusetts, USA

Marcus Roldan
roldan.m@northeastern.edu
Northeastern University
Boston, Massachusetts, USA

Julian Savini
savini.j@northeastern.edu
Northeastern University
Boston, Massachusetts, USA

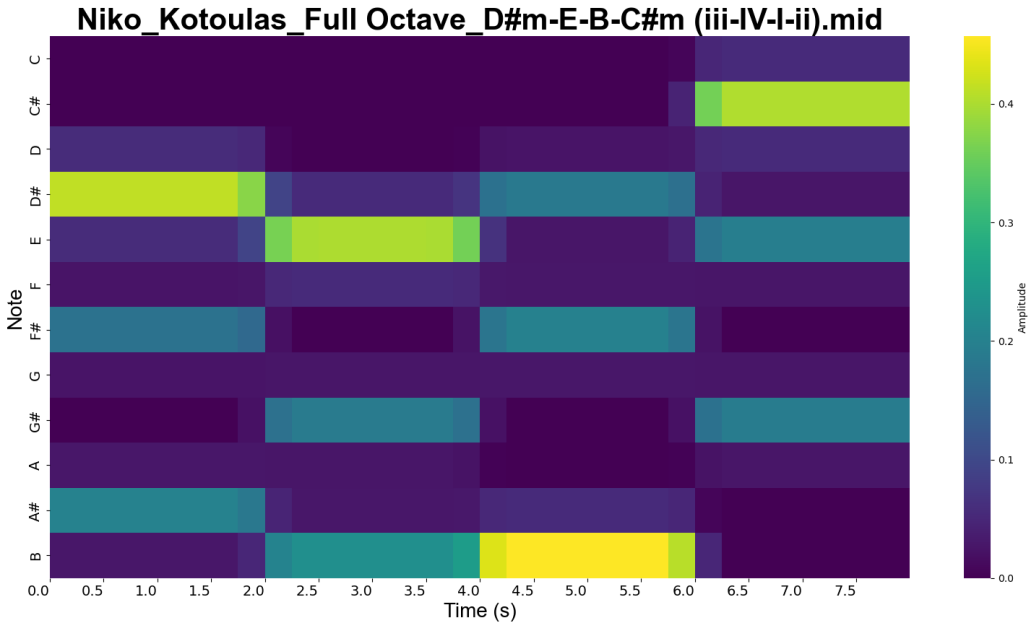

Figure 1: MIDI Sample Chromagram

ABSTRACT

Our project dives into the development of chord recognition/prediction models leveraging Hidden Markov Models (HMMs) in the context of music. Recognizing the growing immersion of artificial intelligence in our day-to-day lives, we wanted to explore if we could use principles from CS4100, and machine learning in general, to create our own robust model to recognize and predict chords, given sequences of MIDI data.

After rigorous initial research, we realized that we would need a chord recognition model in addition to an HMM model for our approach, so we tested a variety of classification and ensemble learning algorithms in addition (Decision Trees, K-nearest Neighbors, Random Forests, SVMs, and Naive Bayes) to find the most optimal.

We trained an HMM model to find the subsequent chord. We utilized chromagrams, which represent the energy distribution of audio signals across different pitch classes, as features for our model. Our HMM model accounted for 36 chord types, including major, minor, and diminished chords. Through extensive training on a dataset comprising 4500 song sequences, and we computed averaged state to state probabilities.

Evaluation was made possible by utilizing a Levenshtein distance formula, a metric commonly used in string similarity analysis. Finally, to demonstrate real-time chord prediction, we integrated our model to a MIDI keyboard controller, proving our model can assist people in real time. Our project offers a promising future in which people could use AI to assist their music writing process.

KEYWORDS

Chord Recognition, MIDI, Chromagram, Machine Learning, Gaussian HMM, Viterbi Algorithm, Classification

CS4100, Spring 2024, Boston, MA, USA
.

ACM Reference Format:
Talia Berdichevsky, Gianna Mattessich, Marcus Roldan, and Julian Savini. . Chord Recognition and Progression Predictor. In *Proceedings of Foundations of Artificial Intelligence (CS4100).* ACM, New York, NY, USA, 4 pages.

1 INTRODUCTION

Our inspiration for chord prediction arose from the plethora of use cases we brainstormed. For instance, it is common for songwriters to hit writer's block, and thus the time it takes to write songs can be drawn out. We do not claim our model can write songs, but we think our predictor can assist composers when they are stuck. Our model can be especially helpful when composing solos/instrumental bridges, when the song will typically exist in a key, but not have the intense repetition of a verse/chorus. Additionally, chord prediction can extend music software capabilities, via interactive music generation, composition assistance [as mentioned], and even real-time accompaniment.

In the universe of music analysis, understanding concepts such as keys, scales, intervals (which is essential for classifying minor, major, and diminished chords), harmony, and rhythm are essential for composers and musicians.

Central to our study is MIDI data, and our manipulation of MIDI data to produce a chromagram. MIDI data is a standardized format for representing notation and their associated sound properties. Chromagrams as described above, are important in representing chords over time, and thus are great input features for music prediction. Additionally, the less noisy data from MIDI input provides improved and cleaner representations of audio, so that those beginning to effectively learn music theory can learn independent of a required instrument.

Our study encountered several challenges in developing an effective model. As we mentioned, it was learned that chord recognition was needed to generate observed states for our HMM model. Secondly, we needed to create an emission matrix for our HMM, which represents the probability of certain pitches given each chord state. Furthermore, we wanted a model that could predict all 36 main chords, and this involved averaging probabilities of reaching all chords across all songs. Solutions will be described in the methodology section.

Our primary goal is to create a model to predict chords, with a relatively decent accuracy. Although we do not expect our predicted chords to always match the actual chords, we do want the distances between these chords to be relatively small. We want our system to be worthwhile, not just for ourselves, but also for musicians who would like to use our program in real-time, as this would make our model not just impressive, but valuable.

2 RELATED WORKS

2.1 Automatic Chord Recognition from Audio Using an HMM with Supervised Learning

Lee and Slaney [1] used a Hidden Markov Model to perform chord recognition of raw audio files. A Pitch Class Profile (PCP) or chromagram was generated from the raw audio files, a dataset of 175 String Quartets from Haydn. The initial chord probabilities, chord transitions, chord covariance matrices, and mu values were calculated from the training data, with the Viterbi algorithm being used to derive the optimal chord sequence with maximal likelihood. Their model achieved 93.35% chord recognition accuracy. There was observed confusion between A-minor seventh and C-major chords, with a hypothesis that in the presence of a G note, the A-minor chord is mis-recognized as C-major because the two chords share the C, E, and G notes, and Lee and Slaney's model groups the A-minor triad and A-minor seventh as a single class.

2.2 Melody Informed Musical Chord Generation Using HMM

Bhular, Peyton, and Xu [3] created a Chord Generator using a Hidden Markov Model using the Viterbi algorithm and a dataset of 1,300 pop songs from the McGill Billboard dataset and 32 of Beethoven's sonatas. Their model revolved around a central assumption that one of the three notes in a triad is the current note in a melody. The transition matrix and initial chord probabilities were learned from the training data, but the emission matrix was calculated from the current input during generation. That is, while the transition matrix and initial probabilities stemmed from all training data, the emission matrix depends on only the given input observations. Inspection of the generated chords revealed more interesting and varied chord generation for classical music than for pop music, with Bhular, Peyton, and Xu hypothesizing that the increased number of chord transitions in classical music than pop music contributed to this observation. Additionally, it was found that pop music chord generation typically returned either the 1 or 5 chords. The authors note that this behavior is not necessarily incorrect, but was less ideal.

2.3 Hidden Markov Models for Chord Recognition

The repository [2] by Caio Miyashiro was developed for the PyCon & PyData conference in Berlin in 2019 and contains an interactive notebook developed for a workshop about using applying Generative AI, and introducing the concept of Hidden Markov Models through their application for chord recognition. The notebook, while intended to serve as an introduction to the concept of HMMs, contains useful examples of Python libraries used to interface with audio and music data, as well as clearly labeled explanations about the music theory required to implement the HMM effectively.

3 METHODS

We had a lengthy initial fact-finding process in order to grasp what properties were needed to train a chord prediction model. First, we created a chromagram, commonly used in audio processing to create a mid-level representation of time in relation to the twelve-tone scale found in Western tonal music. This was then used as input to our models.

For our Hidden Markov Model we had a chord transition probability matrix to represent the probabilities of moving from one chord to another, an emission matrix to represent the probability of an observation belonging to each chord type (or its context/hidden state in the scale), and an initial state probability matrix which indicates the probability of a chord being first in the sequence. We assumed our emission probability matrix followed a multivariate

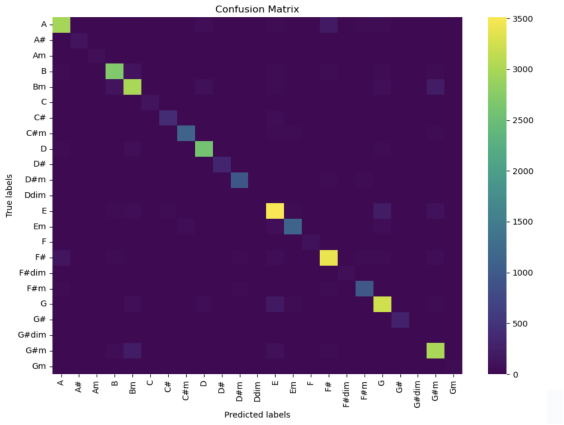

Figure 2: Chord Classifier Confusion Matrix

Gaussian distribution, where each chord in the note energy distribution in the chromagram has a mean and covariance to represent how the note energies vary amongst themselves in a chord.

When beginning to test our trained HMM we realized that the strong assumptions the model makes would be a bit too simplistic for real-life use cases where chords played during music creation with a MIDI don't necessarily have a consistent, sequential state within a chord scale. To account for this fact, we provided our chromagram as input to 2 different models to utilize the capabilities of both. We also found that the skewed dataset was making the model always follow a very rigid sequential pattern based on the data it was trained on. We then tried multiple different classification and ensemble learning methods to identify single chords from the chromagram input, and found the best performing to be a soft voting classifier with random forest and k-nearest classification. We used this output for real-time prediction of chords being played. The underlying hidden states of our HMM were then treated as the likely chord and its role in the scale being played, and we treated our classifier output as observed states.

After both our models were trained, we implemented the ability to input real-time MIDI data, which calculates the note energy distribution at each change in input frequencies and its most likely chord based on outputs from the classifier. Then, after a given number of beats, the Viterbi algorithm and emission matrix is used to infer the most likely hidden chord progression that generated the emissions.

4 EXPERIMENTS/RESULTS

First, we will address our chord classification model. As displayed in Figure 1, we determined occurrences of notes from MIDI data, and from there were able to make a classification for the chord they produced at any instant throughout the piece. For initial chord classification testing, we calculated a rather positive F1-score of 0.89. To further assess our model and find where it may be weaker, we plotted a confusion matrix that addresses the model's individual accuracy on predicting each of the 36 chords (Figure 2).

The confusion matrix in Figure 2 displays the desired diagonal line, showing that the model often makes successful true chord

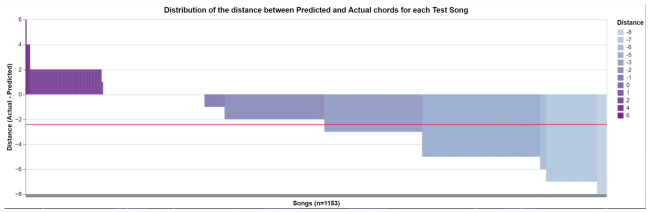

Figure 3: HMM Predictions Distance Distribution

matches. The line is not consistent in color, however, indicating that some predictions are worse than others when it comes to specific chords. In particular, we found that our diminished chord classification needed the most improvement. We believe this is due to the lack of diminished chords in our training data- it was scarce and the lack of diminished chords worsened the model's performance on them, leading to misclassification as seen in Figure 2.

In order to assess our HMM prediction model's performance, we calculated a distance metric measuring semitone and modal distance between any two chords, inspired by Levenshtein distance. Separating the last chord from the start of a test piece, we compared the last true chord to the prediction made by the model. We calculated a resulting mean distance of 2.9 edits between predicted and true chords, equivalent to about 77% percent accuracy for predictions. Figure 3 visualizes the sorted chord distances and mean (red line), displaying a tendency to predict higher chords when making incorrect predictions.

The following are two prediction examples of the last chord sound in a piece, the first being an incorrect prediction and the second being correct.

- **Melody 7 D-Em-G (I-ii-IV)**
  Predicted progression: D, Em, G, C#
  Actual progression: D, Em, G, G
- **Melody 3 Bm-E-G-A (vi-II-IV-V)**
  Correctly predicted progression: Bm, E, G, A, A

Finally, Figure 4 is a live demonstration exemplifying both our work's classification abilities and predictions for future chords. We play a series of chords, automatically recognized by our classification model, and based on the pattern those chords follow, the model makes a prediction for the next chord to come.

5 DISCUSSION/CONCLUSION

While our model is generally accurate in predictions and seems to be of use, there were many lessons learned and things we did not originally account for. Because music is subjective, it is difficult to provide an objective accuracy metric for a chord prediction system. After developing our model, we found many drawbacks to our original approach. Firstly, the strong assumptions of independence and a stationary normal distribution in a Gaussian Hidden Markov Model fail to capture long-term and underlying dependencies that occur in chord progressions. Our choice to use the Niko dataset also created data imbalance because of the lack of certain keys and chords. This significantly skewed our metrics, and led to our models overfitting. If we were to re-approach the problem, we would like to use an RNN-LSTM architecture with a larger and

more balanced dataset. This would likely perform better in a chord prediction task due to its ability to capture more underlying long-term dependencies and chord types. Additionally, we would like to extend the functionality of our system to be more creative, rather than simply predicting the sequences they were trained on, in order to maximize the use of our system for composers hoping to create more unique and interesting musical pieces.

https://youtu.be/VEs2EubxA3I

Figure 4: Live Demonstration of Chord Recognition and Prediction

6 TEAM CONTRIBUTIONS (65 WORD MAXIMUM)

Talia was responsible for preprocessing data, and modifying code to account for all 36 chords. Gianna was responsible for chromagram creation and chord prediction pipeline. Marcus was responsible for creating the transition, emission, and initial state matrices. Julian was responsible for model building and results. All project members were focused on debugging and assisting peers throughout all stages of the project.

7 CODE REPOSITORY

The final code repository can be found in Figure 5. An explanation of the critical files will follow.

7.1 `chroma.py`

The `chroma.py` file outlines various functions used for extracting chromagrams from the training MIDI data. The function `get_chromagram()` uses various helper functions to compute the chromagram for a given song and returns it as a DataFrame.

Additionally, `chroma.py` contains the `get_chromagram_plot()` function which creates a visual chromagram plot for the given song in the training data, an example of which can be seen on page 1 of this report.

7.2 `utils.py`

The `utils.py` file contains utility functions used throughout the process of defining and training our models, including helper functions for calculating the transition matrix, initial chord probabilities, covariance matrices for each chord, as well as functions to predict chords and evaluate those predictions.

7.3 `voting_classify_chord.py`

The `voting_classify_chord.py` file defines the voting classifier used for chord recognition. The Decision Tree, K-Neighbors, and Random Forest classifiers are defined and assembled into the Voting Classifier.

7.4 `midi_classifier.py`

Combines elements of pipeline for new input data. Takes in live midi input and executes model predictions. Every change in active note amplitudes will produce a new prediction for the individual chord. After 16 sequential chords, the HMM model is ran to produce a new chord prediction given the previous chord states.

https://github.com/julianwsavini/CS4100_Final

Figure 5: Project Repository

REFERENCES

[1] Kyogu Lee and Malcolm Slaney. 2006. Automatic Chord Recognition from Audio Using a HMM with Supervised Learning. https://ccrma.stanford.edu/~kglee/pubs/klee-ismir06.pdf. In *7th International Conference on Music Information Retrieval*. ISMIR 2006, Victoria, Canada, 133–137.

[2] Caio Miyashiro. 2019. Hidden Markov Models for Chord Recognition. https://github.com/caiomiyashiro/music_and_science/tree/master.

[3] Sheng Xu, Albert Peyton, and Ryan Bhular. 2018. Melody Informed Musical Chord Generation Using HMM. https://hajim.rochester.edu/ece/sites/zduan/teaching/ece477/projects/2018/ShengXu_AlbertPeyton_RyanBhular_ReportFinal.pdf.