Musical Key Prediction and PCA on Spotify Audio Data Using Fast Fourier Transform

Julian Zimmerlin
Matrikelnummer 6009977
julian.zimmerlin@
student.uni-tuebingen.de

Lennart Slusny
Matrikelnummer 5977176
lennart.slusny@
student.uni-tuebingen.de

Abstract

In this project, the musical keys of piano records are predicted based on their frequency spectrum, calculated using Fast Fourier Transform (FFT). After collecting 30-second audio clips and key labels from the Spotify Web API and applying FFT, we use multinomial logistic regression for multi-class classification and achieve a cross-validation accuracy of 72.8% when predicting the *relative* key (i.e., keys with identical scales are categorized as one class). By visualizing the weights of the classifier, we recover the musical scale that belongs to each key. Furthermore, we provide insight into the systematic errors the classifier tends to make, namely that it sometimes misclassifies keys by a perfect fourth or perfect fifth. In addition, we perform PCA on our piano music dataset, which results in principal components that one can *listen* to, providing a new and interesting perspective on PCA.

1 Introduction

A common method to analyze which frequencies compose an audio signal is the Fourier transform; a reversible mapping from time to frequency space. This process can be seen in Figure 1 where a superpositioned and a concatenated signal of $4, 30, 60$ and 90 Hz sine waves is transformed. Although the temporally varying concatenated signal leads to broader peaks in frequency space and a nontrivial distribution across all frequencies, the spectrum still conveys the dominant frequencies in the overall signal. This justifies our use of the Fourier transform over the whole clip length in the following.

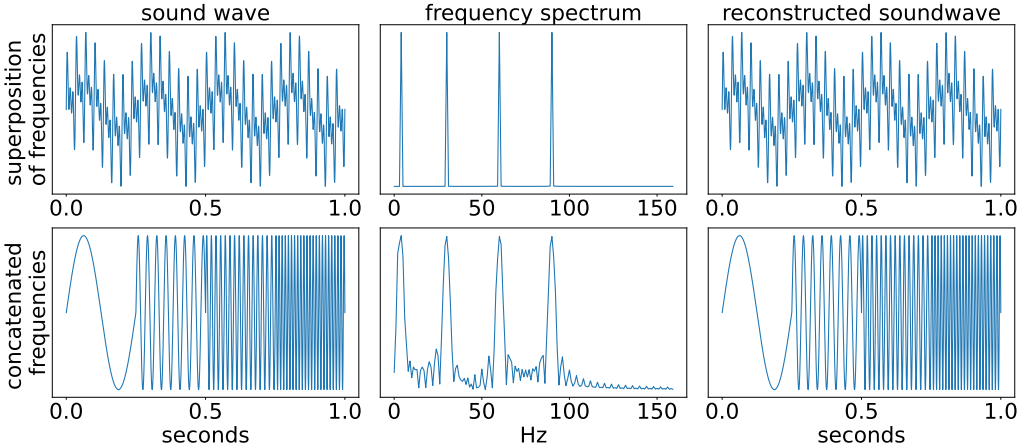

Figure 1: Fourier transform of superpositioned and concatenated 4, 30, 60 and 90 Hz signal

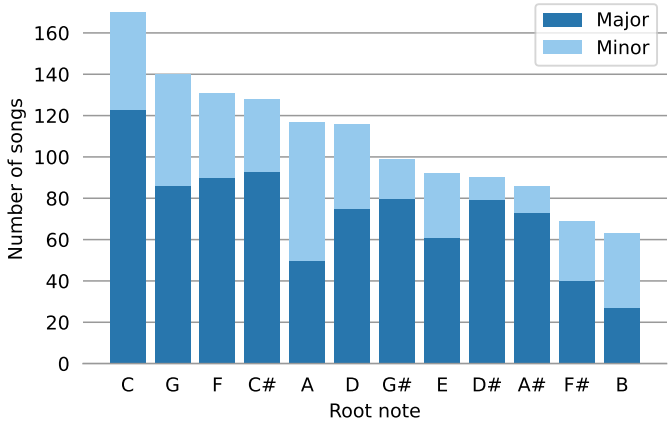

Figure 2: Key distribution in our dataset (n=1301).

2 Data and Preprocessing

To collect our dataset, we manually created a Spotify playlist of 2773 piano recordings, including both classical and modern pieces [1]. We use the Spotify Web API [2] to collect 30 second audio clips of these songs in `.mp3` format along with the `key`, `key_confidence`, `mode` and `mode_confidence` values which we use for the labeling of our dataset. The `key` attribute represents the root note of the song (e.g. C, C#, D, ...) and the `mode` represents the tonality of the song (major or minor). It should be noted that Spotify does not differentiate between enharmonically equivalent keys, so we will use the convention to always refer to the keys using sharps (#) instead of flats (♭). The confidence scores for both values range from zero to one. The distribution of these attributes in the dataset is shown in Figure 2. After filtering out songs where no audio clip or analysis data was available, as well as songs with a `key_confidence` below 0.5, our dataset consists of 1301 songs.

After converting the audio clips to `.WAV` format at a sampling rate of 2000Hz (PCA) or 4000Hz (classification), we compute the Fast Fourier Transform (FFT) of the recordings to get the frequency spectrum. For the classification task, we perform additional preprocessing steps: We convert the complex frequency components into real numbers by taking the absolute values, since the phase information is irrelevant for assessing which frequencies are most prominent in a song. Because the audio signal is real, the absolute values of the frequency spectrum are symmetric, so that half of them can be discarded, which leaves us with 60000 features (4000Hz * 30s / 2). We reduce the dimensionality of the dataset to 72 features by binning the frequencies into groups belonging to exactly one note on the piano each, ranging from *A0* to *B5*, and taking the mean of all frequency components in the group. The interval boundaries are calculated by taking the mean of each pair of frequencies of adjacent notes in equal-temperament tuning, as specified in [3]. Frequencies outside of this six-octave range are discarded.

3 Principal Component Analysis

To explore the frequency space and the reproducibility of the song clips by their frequency spectrum, a principal component analysis was conducted. A reconstruction of Beethoven's 'Für Elise' can be seen in Figure 3. Furthermore, a video [4] is provided to enable the reader to listen to some of the principal components and to assess the quality of 'Für Elise' with increasing number of principal components. Interestingly, the first principal components already contain interesting structure, with individual audible notes and even chords, whereas later components seem less structured. Due to memory constraints, we had to use a smaller dataset size of 959 songs here, which is rather low compared to the 30000 features. Moreover, all songs in the dataset belong to the same genre, with music by mostly a single piano. One may assume that dimensionality reduction on a more diverse dataset would be more challenging.

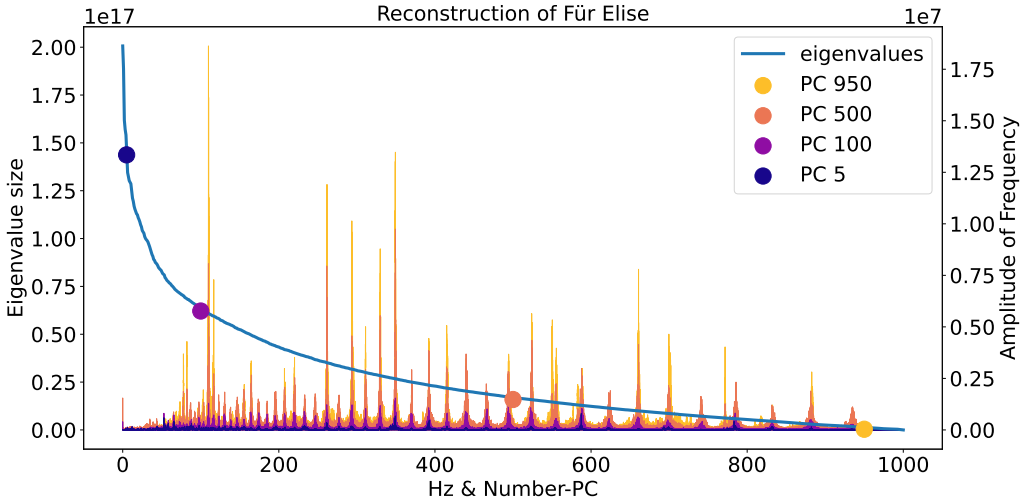

Figure 3: Reconstruction of 'Für Elise' with increasing number of principal components

4 Key Classification

We set out to solve two different variants of the key prediction task:

(1) Predicting the *exaxt musical key* of the song, which means predicting the root note of the song along with predicting major or minor tonality, wich results in a 24-class problem.

(2) Predicting only the *relative major key* of the song, which results in a 12-class problem. Every minor key has a relative major key that is made up of exactly the same notes (for example, A minor and C major). The only difference is the order of the notes in the scale that belongs to the key, which we suspected would make these keys hard to differentiate using only the frequency spectrum. Since musicians also frequently modulate between relative keys, one may consider this evaluation method more fair given our experimental setting, where we use only small snippets of each song.

We used multinomial logistic regression with L2 regularization and the SAGA solver [6] as a natural extension of logistic regression to multi-class datasets. Surprisingly. the solver is important, as all other solvers implemented in `sklearn` perform significantly worse. We also tried binary logistic regression with a one-vs-all scheme to extend to the multi-class case, as well as Support Vector Machines with different kernels, but both alternative methods showed subpar performance. We used 10-fold cross validation repeated 5 times to evaluate the model performance.

Without frequency binning, we obtained a cross validation accuracy of 53.9% on the exact prediction task (due to time constraints, this number was obtained with a single repetition of CV instead of 5 repetitions) and 69.6% on the relative key prediction task. With frequency binning, performance increased slightly to 55.0% and 70.8%, respectively. In addition to improving accuracy, the reduction in dimensionality drastically improved the runtime from more than 9 hours to less than 30 seconds for the full 10-fold cross evaluation with 5 repetitions. Due to the periodicity in the features (i.e., each note being represented 6 times across the different octaves), it is possible to further decrease the dimensionality to 12 features, each one being the mean frequency component of one note across all octaves. With this technique ("double binning"), the CV accuracy further improved to 57.2% and 72.8% for the two tasks.

We plot the confusion matrix of the 12-class problem with double binning in Figure 4. One may notice that a common error that the classifier makes is to misclassify keys by an interval of a perfect fourth or perfect fifth. This is consistent with music theory, since these keys share 7 out of 8 notes with the original key, only adding or removing a single sharp or flat. In the exact prediction task, we observe that minor keys are more often misclassified than major keys (see confusion matrix in [5]), most probably because of the class imbalance in our dataset. Furthermore, the most common misclassification is predicting the relative major key, confirming our initial suspicion of relative keys being hard to differentiate based on the frequency spectrum.

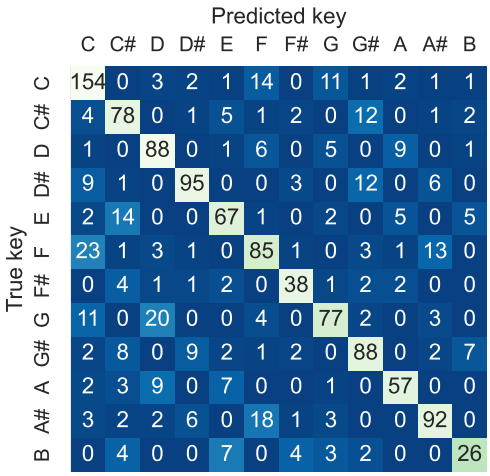

Figure 4: Confusion matrix of multinomial regression with double binning (12x12 weights). Color values normalized by row.

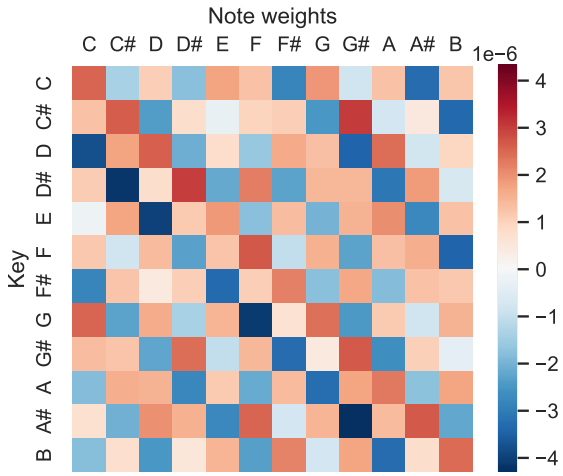

Figure 5: Visualization of learned weights with double binning.

Figure 5 shows a visualization of the learned weights. You can see that the classifier learns which notes belong to which key, since notes that belong to the key have positive weights, whereas other notes usually have negative weights. Due to the shared structure between the different keys, it may be possible to further improve the model by implementing weight sharing between keys to reduce the total parameter count to 12 (instead of 12 per class).

5 Limitations and Outlook

A limitation of our approach is that the labels we obtain from Spotify's API are far from perfect. We traded off dataset size and label quality by using only songs with a `key_confidence` greater than 0.5 and checked by hand[1] that the labes for these songs are usually alright[2]. However, there are still some false labels; in particular, relative keys also seem to be a problem in Spotify's key detection algorithm. Thus, our performance measures really only measure how well we approximate Spotify's algorithm instead of "ground truth" accuracy.

An idea for future work would be to not just compute the FFT of the whole song and instead bin the song into small time intervals and perform FFT on each of them. Models designed for sequential data, like RNNs, could then analyze the harmonic progression of the song for more accurate key prediction (for example, a progression from the dominant to the tonic chord of a key should be a highly potent indicator). This may also help alleviate the problem of relative major/minor keys.

References

[1] https://open.spotify.com/playlist/2J1TAsLuUuwH0iqjlw3OVj. [05-02-2022].

[2] https://developer.spotify.com/documentation/web-api/. [05-02-2022].

[3] https://pages.mtu.edu/~suits/notefreqs.html. [05-02-2022].

[4] https://www.youtube.com/channel/UCxs13U6aHAxavfIQ3d4ma8Q. [06-02-2022].

[5] https://github.com/lennytubby/KeyPredictionbyFrequency. [05-02-2022].

[6] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

[1] For classical pieces, the true key can be easily found online and is often even contained in the title.

[2] In particular, the `key_confidence` does not seem to represent the probability of the label being correct.