Detecting Musical Key with Supervised Learning

Robert Mahieu
Department of Electrical Engineering
Stanford University
rmahieu@stanford.edu

***Abstract*—This paper proposes and tests performance of two different key-estimation system architectures based on supervised learning principles and music fundamentals. The systems take as input an average chroma feature vector representation of a query song and returns both the estimated mode and tonic note, together representing the key of the piece. The systems were both trained using features and metadata from the Million Song Dataset. Generalization error during training was typically quite low (less than 25%), and final tests on the dataset indicated that both architectures may be capable of successful classification over 80% of the time. Perfunctory tests on a few extra songs outside the dataset resulting in the second system performing significantly better, correctly classifying 7 of 12 songs with relatively difficult/obscure keys.**

I. Introduction

When building a piece of music, to ensure that the pitches used will flow and exist in consonance with one another, the musician must have an understanding of the underlying key of the piece. This "key" is based upon a chosen underlying scale from which fundamental relationships between certain notes give rise to sets of intervals and chords that quite simply sound good together. Within the context of this project, the key of a piece will refer to both the mode (minor or major) and the tonic note (C, Db, D, Eb, E, F, F#, G, Ab, A, Bb, or B) of the fundamental scale.

Due to this property of music, one of the most valuable pieces of information for a DJ or music producer when mixing or remixing songs is the key of the piece they are working with so that they are able to piece together tracks that sounds good together. Furthermore, it is of tremendous added benefit if they do not have to spend time manually labeling each and every song in their library, which can commonly include hundreds of thousands of songs.

Within the industry, there are a handful of software packages available that advertise the ability to automatically detect the key of a song, however there exists a relatively large trade-off between accuracy and price. For example, a 2014 study by DJ TechTools found the most accurate package to be *Mixed In Key*, with a hefty price tag of $58, which gave an accuracy of 95% on their test dataset [1]. The best free software was found to be *KeyFinder*, with an accuracy of 77%.

As a result, the goal of this project is to use supervised learning techniques to develop a classification system to label any given song with its correct key and ultimately produce a free software package that allows anyone to analyze their own music library with high rate of success. Ideally, a testing accuracy over 77% would be rather remarkable.

To achieve this goal, two system architectures are investigated: one which first classifies mode and then classifies key, and another which first classifies the notes used in the underlying scale then classifies mode and key together. The input to the system is always a 12-length average "chroma" vector which represents the relative strength of every pitch in the chromatic scale throughout the entire song.

II. Related Work

Previous approaches to detecting musical key can be essentially grouped into two categories: one in which the algorithm attempts to develop a "tonal profile" of the song then match it against a set of pre-defined tonal profiles for each key, and another in which hidden Markov models (HMMs) are learned for each key and then the most likely key is determined from the HMM given an input chroma sequence.

The tonal profile approach stems from psychology research originally presented in 1982 by Krumhansl and Kessler in which they developed a measure of the musical importance of each note to each key [6]. To do this, they had a musician play seven notes of a scale followed by a random note and instructed participants to rate how well the last note fit musically with the scale. By averaging the ratings throughout many trials and different scales, the study constructed what they called a tonal profile for each key which contained relative importance weights for each note. The profiles resulting from this study, dubbed the "KK-profiles," were found to be structurally the same between keys in the same

mode–just transposed up or down accordingly. However, between modes the structure was somewhat different. As such, two fundamental profiles were given for each mode which would then be used to represent any key.

To then do key-detection, an input song is pre-processed into a chroma vector representation, then a correlation function, such as the cosine similarity, is used to find which key's tonal profile is most similar. However, this method has been found to be very sensitive to the exact profile weightings used. Slight variations of the KK-profile have been found to produce significantly better or worse classification results [2].

The HMM approach first processes each song into a sequence of chroma vectors over time. Here, the characteristics of the modes are learned by training two HMMs on labeled data, then 24 HMMs corresponding to all the various keys are derived from the trained models [7]. The key of the input song is selected corresponding to the HMM that gives the highest likelihood of the chroma sequence. To my knowledge, this is the only other machine learning approach developed for detecting musical key.

Both of these approaches suffer from common mismatches to keys a 5th up or down from the true key, due to the close similarity in pitches between the underlying scales.

III. Dataset

This project makes use of the *Million Song Dataset* [3], which is a massive collection of features and metadata for one million songs, provided by The Echo Nest. The data here of particular interest are the chroma features and metadata for the key and mode of each song.

Note that due to the large size of the entire dataset (>300GB), this project was only able to use a 10,000 song subset.

Unfortunately, the key and mode metadata is not actually ground truth. Instead, each song is attributed a confidence metric $c \in [0, 1]$ indicating how confident the dataset is in the classification given. This makes training and testing a bit more difficult. To ensure enough data was available for training, only data above a threshold of $c = 0.5$ for both key and mode were used, which limited the data to a subset of 3729 songs.

IV. Feature Selection

Because chroma features provide a representation of the relative strength of each pitch class present at each time window in a song, I choose to use features based on this data for the key-classification system. These give insight on the types and frequency of pitches used in the song and should therefore directly relate to the key. Accounting for the variable lengths of songs and avoiding overly-complex feature representations, I choose to average the chroma vectors given throughout the length of the song. This produces a 12-length feature vector that is then used as input to the system. Figure 1 below demonstrates this process for one example song:

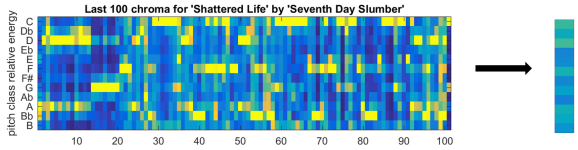


Fig. 1: Example of feature creation. *Left)* Last 100 chroma features generated from the input song. *Right)* Average chroma vector to be used as final features.

V. Methods

Two system structures were investigated in this study. The two are illustrated in Figure 3.

A. Architecture #1

The first architecture (Figure 3a), is based off the tonal profile formulation discussed in Section II and seeks to instead construct a model of tonal *regions* using supervised learning. This method first seeks to produce classification boundaries between minor and major modes, based on the understanding that tonal profiles differ significantly in structure (weightings) between modes. Once the mode of the song is classified, the system then attempts to classify the actual pitch class (key), with the belief that each tonal profile within a given mode is simply some transpose of the others and therefore the tonal regions should be separable.

For both layers of classification, both multi-class support vector machine (SVM) and multinomial logistic regression (softmax regression) models were tested. As will be detailed further in Section VI, experimental results indicate that the SVM model works best for classifying the mode, while the softmax model works best for classifying the pitch class.

The SVM model is trained to distinguish between two classes $y \in \{-1, 1\}$ by minimizing the cost function:

$$J(\alpha) = \frac{1}{m} \sum_{i=1}^{m} L_{hinge} \left(\sum_{j=1}^{m} \alpha_j K(x^{(i)}, x^{(j)}), y^{(i)} \right)$$

Which represents the average loss over all m training examples using the hinge loss function defined as:

$$L_{hinge}(z, y) = [1 - zy]_+$$

The kernel function $K(x, z)$ is chosen to be the radial basis function:

$$K(x, z) = \exp\left(-\frac{1}{2}\|x - z\|^2\right)$$

The cost function $J(\alpha)$ is iteratively minimized using gradient descent methods.

In order to control overfitting, regularization was employed by adjusting the box constraint. Another technique called standardization was also used which centers and scales each dimension of the training data by the weighted column mean and standard deviation.

Note that in order to use the SVM model—which fundamentally only distinguishes between two classes—with multiple classes, one SVM is trained on each class vs. all other classes which then returns a score indicating how likely the input is representative of that particular class or not. This formulation therefore allows us to use a collection of binary classification SVMs to classify more than two classes by selecting the class corresponding to the highest score among all returned by the SVMs.

The softmax regression model is directly trained on K total classes by using maximum likelihood estimation on the likelihood function:

$$L(\theta) = \prod_{i=1}^{m}\left(\prod_{k=1}^{K} P(y^{(i)} = k|x^{(i)}; \theta)^{1\{y^{(i)}=k\}}\right)$$

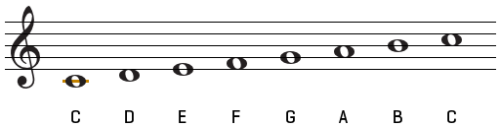
Where the probability term is:

$$P(y^{(i)} = k|x^{(i)}; \theta) = \frac{\exp(\theta_k^T x^{(i)})}{1 + \sum_{j=1}^{K} \exp(\theta_j^T x^{(i)})}$$

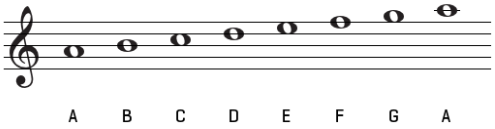
This produces $K-1$ classification vectors θ_i (represented altogether in the above representation by the symbol θ without subscript). Note that $1\{x = y\}$ represents the indicator function. We iteratively optimize using gradient descent methods on the log of the likelihood function.

B. Architecture #2

The second architecture tested in this project (Figure 3b) was formulated using the fact that each key is based on some underlying set of eight notes, regardless of whether or not their particular usage is representative of the same scale. For example, the C major and A minor scales, shown in Figure 2, contain the same notes, but actually represent different keys due to the different tonic notes and corresponding chords. The hypothesis for this formulation is that it may be possible to first classify an input into one of 12 dual-key classes which share the same eight notes, then from there attempt to perform a more intricate classification to determine which of the two possible keys the song truly represents.



(a) C major scale



(b) A minor scale

Fig. 2: Illustration showing how two different scales may contain the same pitches.

TABLE I: Major and minor keys which contain the same notes

Major	Minor
C	A
Db	Bb
D	B
Eb	C
E	Db
F	D
F#	Eb
G	E
Ab	F
A	F#
Bb	G
B	Ab

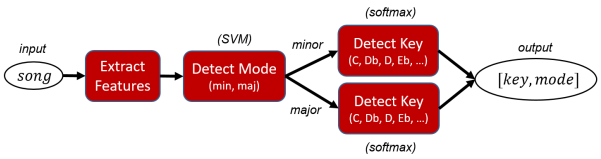
To do this, a softmax regression model is first trained on the 12 classes which represent keys containing the same eight dominant notes. Each includes one major and one minor key, summarized in Table I. Next, one logistic regression model (softmax with only two classes) is trained for each of these dual-key classes in an attempt to distinguish the two possible keys within each. This produces a total of 12 new classifier models.

Note that due to the limited amount of data available, an SVM model classification model is not considered for this second approach. Preliminary results from testing architecture #1 indicated that softmax should be used in this case in order to best avoid overfitting.

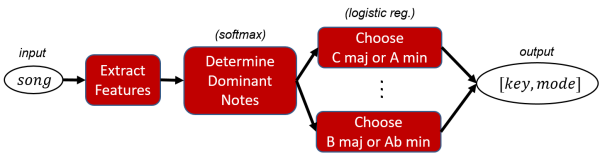
VI. Results

A. Architecture #1

All models were trained and tested using only data possessing a confidence value above 0.5. This provided



(a) System architecture #1



(b) System architecture #2

Fig. 3: Illustrations of the two system architectures investigated in this project.

TABLE II: SVM vs softmax performance comparison for Arch. #1

Classifier	Model	Train Error (%)	Test Error (%)
Mode	SVM	4.885	14.504
	softmax	24.257	25.424
Major Mode Key	SVM	6.615	37.135
	softmax	16.981	25.472
Minor Mode Key	SVM	6.617	21.111
	softmax	17.955	21.389

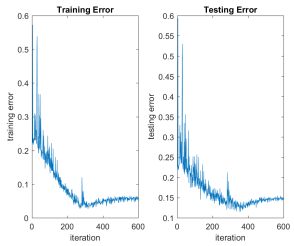


Fig. 4: Error per training iteration for Arch. #1 SVM mode classifier. *Left)* Training error. *Right)* Testing error.

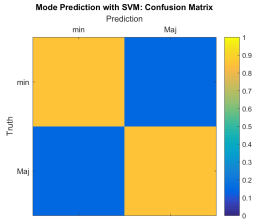


Fig. 5: Confusion matrix for Arch. #1 SVM mode classifier (normalized over columns).

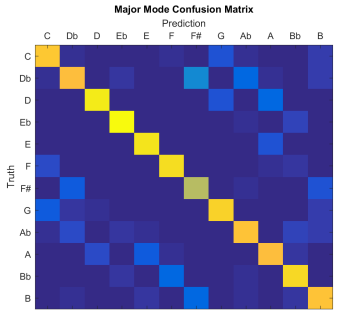


Fig. 6: Confusion matrix for Arch. #1 softmax major key classifier (normalized over columns).

2881 major key and 848 minor key songs, for a total data subset of 3729 songs. Training was performed on a randomly ordered selection of 75% of the data, while a 25% cross-validation set was left out to be used for testing.

A comparison between the results for the SVM and softmax regression models for each classifier in the system are shown in Table II. The results indicate that for the mode classifier (the first layer in the system), the SVM performs significantly better on the test set, misclassifying over 10% less examples. Using these results, the SVM is chosen as the model for mode classification. Note that the SVM was trained several times with different regularizing box constraints and the optimal was found to be a value of 1.0, which was then used for all the results reported in this paper. The most significant influence of this regularizing adjustment is that the underrepresented minor mode data is effectively given more importance relative to the more common major mode data, allowing both to be classified equally well. This is evidenced by the equivalent colors in each column of the confusion matrix in Figure 5.

On the other hand, while the test error is roughly equal between the SVM and softmax results for major mode key classification, the softmax model performs significantly better for minor mode key classification—again with over 10% less misclassifications on the test set. Due to this result, the softmax model is chosen as the model for both major and minor mode classification. The confusion matrix for the trained model for major key classification is given as an example in Figure 6.

B. Architecture #2

The second system architecture was trained using the same data subset and cross-validation testing technique as the first. Training of the dominant note dual-key softmax classifier resulted in a training error of 25.706% and a testing error of 26.073%. The corresponding confusion matrix is shown in Figure 7. Training the second-layer logistic regression key classifiers typically resulted in testing error between about 6 and 10%, a quite remarkable and endearing result. The confusion matrix for the C major vs Ab minor model is shown in Figure 8 as one example.

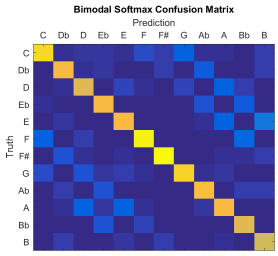


Fig. 7: Confusion matrix for Arch. #2 softmax classifier between dual-key classes (normalized over columns). Labels refer to major key of class.

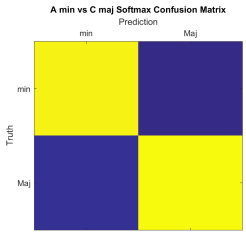


Fig. 8: Confusion matrix for Arch. #2 logistic reg. classifier between C major and Ab minor (normalized over columns).

C. Comparative Performance

To fully test and compare each system architecture, all data from the dataset with mode and key confidences above various thresholds were sent completely through both systems and compared with their labels. Results from this are shown in Figure 9.

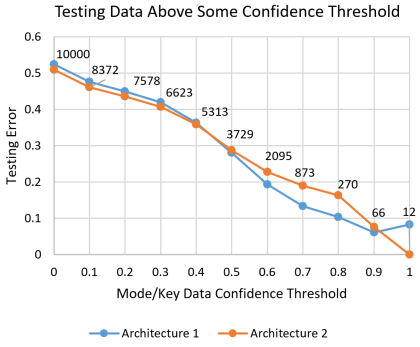


Fig. 9: Performance comparison of system architectures. The number of valid data tested at each threshold is shown next to each point.

While these results appear to imply that both systems perform just about equally well, I was able to run a handful of tests with 12 randomly selected songs from the DJ TechTools survey [1] of which ground truth keys are given. Note that most of these were in relatively uncommon [4] minor keys (e.g. G min, C min, F min) characteristic of obscure electronic music. The MATLAB Chroma Toolbox [5] was used to extract chroma features from the songs. In these tests, summarized in Table III, architecture #2 performs significantly better.

VII. Discussion

The results given in Section VI illustrate that both architectures were quite capable of low generalization error during training. Figure 9 appears to indicate that both systems are actually quite successful in estimating the correct key of music in the dataset, as both attain testing error rates below 20% when confidence on the key and mode data is high ($c \geq 0.6$). This is a percentage comparable to the error of the best free key-detection software available [1].

TABLE III: Results on 12 songs from DJ TechTools survey.

Architecture	Correct (%)	Off by 5th (%)	Off by mode (%)
#1	0.000	8.333	25.000
#2	58.333	16.667	0.000

However, definite conclusions become difficult to make after viewing the results from testing on new data outside the dataset (Table III). Still, it is clear that system architecture #2 performs very well regardless, since even on the new data it is capable of perfectly estimating the key of 7 of the 12 songs. It is important to note, though, that all but one song in the newly tested data were in somewhat obscure minor keys—keys which were relatively underrepresented in training. Therefore the results should not necessarily be considered representative of the systems' ability to correctly classify all keys. This makes the results from architecture #2 all the more impressive. It would be very interesting to do more testing in the future on a larger variety of keys, though reliable truth data would need to be acquired.

Interestingly, even the systems developed in this project appear to still have difficulty with keys differing by a 5th, which is evident by the bright off-diagonal strips in the confusion matrices of Figures 6 and 7. However, it is unclear if this is actually an artifact of the imperfections of the training data. Future training using higher confidence (or ideally true) data should be done to determine if this alleviates the issue at all.

VIII. Conclusion

This paper proposes two key-estimation system architectures based on supervised learning principles and music fundamentals which are able to make predictions using a chroma feature vector representation of a query song. Though data from the dataset was somewhat unreliable, results appear to at least indicate that architecture #2 performs very well, achieving a total accuracy estimated to be somewhere above 58.333%.

References

[1] "Key Detection Software Comparison: 2014 Edition." *DJ TechTools*. N.p., 14 Jan. 2014. Web. 20 Nov. 2016.

[2] Sha'ath, Ibrahim. *Estimation of key in digital music recordings*. Diss. Masters Thesis, Birkbeck College, University of London, London, UK, 2011.

[3] Bertin-Mahieux, Thierry, et al. “The million song dataset.” *ISMIR*. Vol. 2. No. 9. 2011.
[4] Eliot Van Buskirk. “The Most Popular Keys of All Music on Spotify.” *Spotify Insights*. Spotify, 2015. Web. 16 Dec. 2016.
[5] Ewert, Sebastian. “Chroma Toolbox: MATLAB implementations for extracting variants of chroma-based audio features.” *Proc. ISMIR*. 2011.
[6] Krumhansl, Carol L., and Edward J. Kessler. “Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys.” *Psychological review* 89.4 (1982): 334.
[7] Peeters, Geoffroy. “Musical key estimation of audio signal based on hidden Markov modeling of chroma vectors.” *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. 2006.