# Group 35: Musical Key Recognition using a Hidden Markov Model

Jordan Barkai[1] and Pieter Dekker [2]Thijs Havinga [3]Jeroen Overschie [4]

[1]Kapteyn Astronomical Institute , University of Groningen, s3972097
[2]Bernoulli Institute for A.I., University of Groningen, s2546736
[3]Bernoulli Institute for A.I., University of Groningen, s2922924
[4]Bernoulli Institute for A.I., University of Groningen, s2995697

***Abstract*— Musical keys can be thought of as the foundation on which music and songs are created. The development of an automated technique to recognise the overall key of a musical recording is driven by the need in the production and mixing of music. In this paper Hidden Markov Models (HMMs) are used to detect the key of tracks retrieved from Spotify's web API. The relative strength of each pitch in different segments of a track, represented as the chroma vector, was extracted for about 10,000 tracks. The HMMs models were trained for each mode, major and minor, and shifted to create one for each key. Two model states were chosen for the HMM models to optimize the validation error rate as well as the computational time. Additionally, a naive approach was created using cross-correlation. These two methods were compared using 10-fold cross-validation on various sized data-sets. While both models performed better and stabilized further with an increase in the number of tracks, the HMMs out-performed the naive method. A brief investigation of the confidence level associated with the labelled keys found a decrease in validation error for tracks with a higher confidence. It is therefore recommended to further explore the cause of the key confidence and to what extent it affects the training of the models.**

***Index terms*— Chroma vector, Musical key, Hidden Markov Model**

## I. Introduction

Musical keys can be thought of as the foundation on which music and songs are created, an underlying scale of which most notes that comprise it fall within. The key is fundamental for the creation of melodies and harmonies and can be put simply as the combination of chords that sound well suited for each other. Considering the 24 main *chromatic* keys which can be distinguished by the human ear, this paper aims to achieve the same recognition by machine.

The development of an automated technique to recognise the overall key of a musical recording is primarily motivated by the production and mixing of music and the significant time saved by removing the need for manual labelling. Previous attempts include matching developed tonal profiles to those predefined for each key [8] and using hidden Markov models (HMMs) [10]. In this report the latter is used by training an HMM for each key and choosing the key which corresponds to the HMM with the highest likelihood of an inputted "chroma" vector which is defined as the "relative strength of every pitch in the chromatic scale" [9] of the piece. This is explained in more detail in Section II.

Both the chroma vectors and the musical keys were extracted from the Spotify web API [16]. This information was retrieved for approximately 10,000 tracks by taking their track IDs from the Million Song Playlist data-set [4] and using them to collect their audio analysis from the Spotify web API.

In addition to HMMs, a naive analytical approach based on musical knowledge will be investigated for comparison. This is elaborated upon in Section II and used to compare to our machine learning approach to help support or refute the advantage of using HMMs to solve this problem. The results and success of the task are shown and

discussed in Section III and are followed by the conclusion in Section IV.

While there has been extensive work using this technique, the biggest challenge to overcome is the similarity in scale pitches which creates potential mismatching. It is therefore expected that the trained algorithm will be more accurate for "simpler" songs with a more pitch that dominates the chroma vector strongly. This project aims to achieve a decreased validation error with supervised key recognition using HMMs when compared to the previously mentioned naive method.

## II. Method

### A. Data-set

Music can be represented as a sequence of amplitude samples which have varying frequencies and can be played back to produce the music they were recorded from. The dimensionality of such recordings is very high, for example, an MP3 encoded song, consists of a stream of frames, with each frame containing a number of samples. The actual feature density per second for audio recordings depends on the sample rate, and the lowest possible sample rate for MP3 is 8,000 Hz [7]. This means that the lowest feature density per second for MP3 is 8,000 features. The changes from sample to sample and the nuances in sound that are a result of it are very important for the playback of audio. However, for the analysis of that audio this accurate representation is overly explicit and instead for the purposes of this project we used a higher-level representation of the audio data which focuses on dominant tones lasting up to several seconds.

We know that a key consists of a scale of harmonic tones, where the scale for each key is a subset of all the musical tones. Music composed in a certain key will contain these tones in a far greater ratio than any of the other tones. Knowing that each note is produced at a certain pitch, and using time-frequency analysis [13], it is possible to obtain a time-series of pitch strengths of a piece of music, which is expected to closely relate to the musical key the piece of music was written in. An analysis of the pitch strengths of a portion of a piece of music is sometimes called a chroma vector. A chroma vector is a profile of the 12 pitch classes showing the relative strength or dominance of each pitch in the chromatic scale [9]. For example, a C Major chord would be represented by a chroma vector profile dominated by the C, E, and G pitches. It is a 12-dimensional vector containing the normalized log-power of the pitch frequencies for each pitch. We expect such a time-series of chroma vectors to be a good observation sequence to be used in HMM model training and evaluation. Therefore, we construct a labeled data-set of tracks. The data-set will contain, for each track, the key, the mode, and a time-series of chroma vectors.

Spotify's Web API[1] [16] provides audio analysis data of tracks, among which are such features as key, mode and an quantitative analysis of the pitch strengths. Each track is divided into "segments" which have consistent sounds and the audio analysis of each segment contains a corresponding chroma vector. Since Spotify identifies tracks by track-id's, we began by creating a list of track-id's using the Spotify Million Playlist data-set [4]. The Spotify API was then queried to obtain the key and mode for each track-id. We then stored all tracks received from the Spotify API, with the condition of having less than a certain amount of tracks with the combination of key and mode for that track, so as to construct a data-set without bias to a certain key or mode. Since we are able to get chroma vectors from the Spotify API directly, most of the pre-processing that could be done is already done. The series of chroma vectors we receive from the Spotify API are all accompanied by a duration. We will trim the series of chroma vectors so that the sum of the durations is the smallest possible number time greater than 20 seconds. For a better understanding of the chroma vectors for each track, an example plot has been animated for a track from Spotify's API, see Figure 1.

[1] Although the details are not explicitly mention anywhere, the analysis data provided in the Spotify API is created with Echonest software [15].

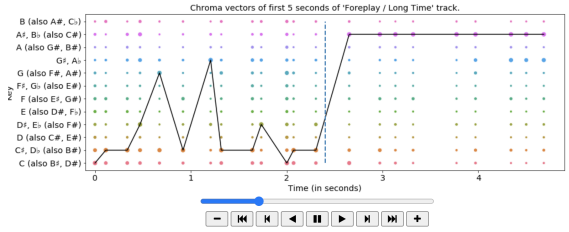


Fig. 1: A screenshot of an animation used to visually explain the chroma vectors of Long Time by Boston. The points represent the different pitches as a function of time, with the size of the point representing the strength of that pitch. The plotted line runs through the maximum pitch as a function of time.

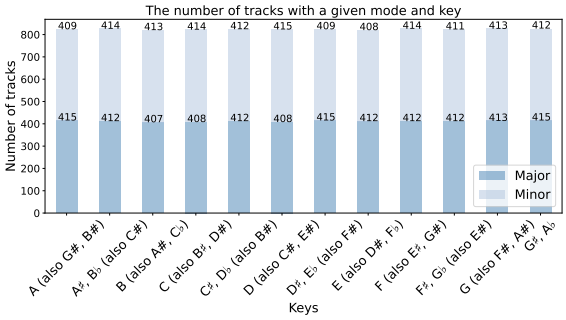


Fig. 2: Distribution of musical key and mode for the data-set, demonstrating a balanced data-set with an equal amount of tracks in each mode and key.

In our attempt to do this for 10,000 tracks, we ended up with 9886 tracks, due to errors during the fetching process. The smallest amount of tracks for any key-mode combination is 408, and the largest amount for such a combination is 415. Considering Figure 2 we deem these amounts close enough together for the data-set to be called balanced.

The labelled key and mode of each track came with an associated confidence, a float between 0 and 1. For the simplicity of this project these labels are assumed to be correct for the initial comparison of the naive and HMM methods. However, looking at Figure 3 it is clear that around 72% of the tracks have confidence levels below 0.4, and therefore an investigation of the effect of filtering by the confidence level was explored by running the final models on various different confidence lower bounds.

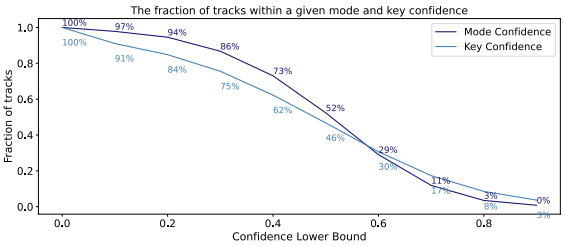


Fig. 3: The fraction of tracks within different key and mode confidence lower bounds. The red line and writing shows the mode confidence bounds and the corresponding percentage of tracks within these bounds while the blue line and writing shows the key confidence bounds and the corresponding percentage of tracks within these bounds

To test the validity of the trimming operation, 1,000 tracks were retrieved and the average entropy was used for comparison to the trimmed versions. The shannon's entropy, $H$ of each chroma vector, a discrete random variable $X$, is defined as:

$$H(X) = \sum_{i=1}^{12} -x_i \log_{12} x_i \qquad (1)$$

where $i$ is the pitches in each chroma vector and $x_i$ is the strength of the pitch.

Since the entropy represents the balance of pitch representation for chroma vector, if $H(X)$ tends to 0 the chroma vector is very imbalanced. Therefore the smaller the entropy averaged over all the segments of a track, the more dominated it is by a single pitch, making the overall key more recognisable. As can be seen in Figure 4, by trimming the songs to the first 20 seconds the entropy is reduced, meaning that there is a more dominant pitch in the chroma vectors, while the distribution for the sample is preserved. This means that besides the computational benefit of trimming the audio analysis, the chroma vectors will be easier to identify a key from, while still representing the full song.

### *B. Learning architecture design*

#### *1) Naive Method*

A naive method was first implemented for a simple key estimation of each track. First the average chroma key of each track was

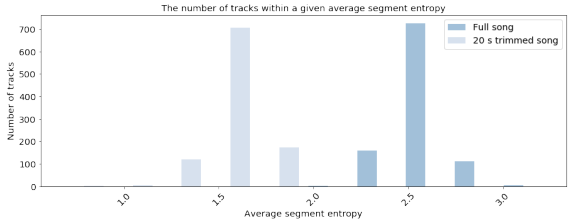


Fig. 4: The distribution of the average segment entropy for each track in a sample of 1,000 tracks. The blue shows the full track while the orange shows the track trimmed to the first 20 seconds.

computed and weighted by the duration of each segment. These were then compared with each scale's notes using a cross-correlation. This method is based on the fact that each key can be characterized by a set of 7 tones that are characteristic for that key. As an example, the key *C major* is characterized by the tones $\{C, D, E, F, G, A, B\}$ [3]. In chroma vector space, these tones can be represented as $h_{Cmaj} = [1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1]$. The correlation between this kernel and any chroma vector, set of chroma vectors, or weighted average of chroma vectors, can give a naive approximation of the likelihood of these vectors to belong to any given key. Note that each major key has a *relative* minor key with the same kernel. For example, $h_{Cmaj} = h_{Amin}$. This method is therefore not able to distinguish between major and minor keys. We will use this naive method as a baseline to evaluate the results of our machine learning method.

*2) Hidden Markov Models*

This was followed by an approach of training Hidden Markov Models (HMMs). An HMM is an augmentation of the Markov chain, with the assumption that the future state of a sequence is dependent only on the current state and nothing prior. However, when there exists a sequence of hidden or latent states which are not observed, HMMs can be used to relate these to a sequence of observations [12].

According to [14], HMMs can be characterized into three different kinds of problems: a likelihood problem, a decoding problem and a learning problem. The last approach is the machine learning approach that is used for training the model in this project, which learns the parameters of the HMM, the state transition probabilities, the emission probabilities and the initial state distribution, given an observation sequence and set of HMM states. The likelihood estimation approach for HMMs is then used to classify new samples. The most well-used algorithm in attempt to solve the learning problem is the forward-backward, otherwise known as the Baum-Welch algorithm [5]. The Baum-Welch algorithm is a special case of the Expectation-Maximization (EM) algorithm which aims to find the HMM parameters which result in a model that makes the observations maximally likely.

For our application, we adopted an approach proposed in [12]. First, we transposed all training sequences such that they could be re-labelled as either *C major* or *C minor*. By applying a circular permutation (circular "shift") to each chroma vector in a track, its key label changes. For example, applying a single circular shift operation on a set of chroma vectors with an expected key of *D major* results in a set of chroma vectors with an expected key of *C# major*. A repeated application of this operation allows us to transform each training sample to a sample with an expected key of *C*, either major or minor since the mode can not be changed by circular permutation. For a more detailed explanation of the application of the circular permutation, see Appendix I. Training two HMMs to maximize the expectation of these samples with key *C*, resulted in two models that, when presented a new test sample, yield a high *likelihood score* for samples in key *C*, and lower scores for other samples.

To recognize keys other than *C*, we could have shifted each test sample to each of the 12 possible keys and determined the number of shifting operations with the highest likelihood and thus the key with the highest likelihood. This is rather inefficient however, and what we did instead is duplicate the two trained models for the *C* key 11 times each, and then transposed the weights of the duplicated models, each representing a different key by using a different number of shifting operations. Now, the model that

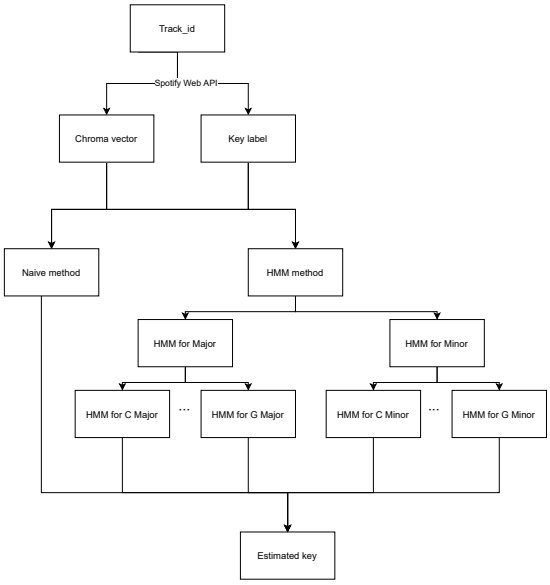


Fig. 5: Architecture Diagram of the two modelling approaches and the overall machine learning pipeline.

yields the highest likelihood score on the original test sample represents the estimated key.

Training only two models which are then replicated to a final 24 models has several advantages. It is much faster to do, saving time. The original two models were trained on a much larger data-set generalized over all different keys, so no model for any particular key can incur a bias towards anomalies that occur only in that key. In other words, this approach also prevents overfitting. Note that the mode was also used in the key estimation from the chroma vectors of each track.

The python package *hmmlearn*, which closely follows the scikit-learn API [11] was used to implement the HMM models, where the distribution of the observations was assumed to be Gaussian. The architecture of this pipeline can be seen in Figure 5.

### C. Hyper-parameter Optimization and Quality Evaluation

Working with data of high dimensionality creates the issue of needing an accurate estimate of many parameters for the emission model, which in turn requires a lot of training data. While cross-validation is the most popular technique used for quality evaluation and hyper-parameter optimization [6], it suffers from its computational expense. Therefore the Peregine HPC cluster of the University of Groningen [1] was used to run 10-fold cross-validation on variations of the model. While the number of iterations was chosen to be 100, high enough such that the model can early stop, a number of experiments were run with varying numbers of model states for the HMM, on a 10-fold setup. This way, the behavior of this hyperparameter with respect to the prediction accuracy can be evaluated, allowing us to pick the optimal setting given our prediction task.

## III. Results and Discussion

The number of optimal states for the HMM model was found by performing 10-fold cross-validation on various different number of states, see Figure 6. It was clear that a choice of two states results in the least validation error while maintaining stability in its generalization capacity and minimal computational expense. While the validation error rate appears to drop at 13 states, the time taken to compute all the folds is over 1,000 minutes making it computationally impractical.

Comparing the results from the 10-fold cross-validation of the naive method with the HMM method it is clear that even from as little as 100 tracks, the HMM method out-performs the naive method. Additionally, as the size of the data grows, the gap between them increases as well as the stability of both models, however, so does the computational expense, see Figure 7. A further investigation of the validity of the original assumption that the labelled keys were correct was carried out by running the 10-fold cross-validation for the 2 state HMM method for various different lower bounds on the key confidence. The results can be seen in Figure 8 which shows a drop in validation error rate of about 15 percentage points when comparing the full data-set to that with a minimum key confidence of 0.8. Additionally, the variation in the error increases with the minimum key confidence, but note that this is likely a result of the subset decreasing in size.

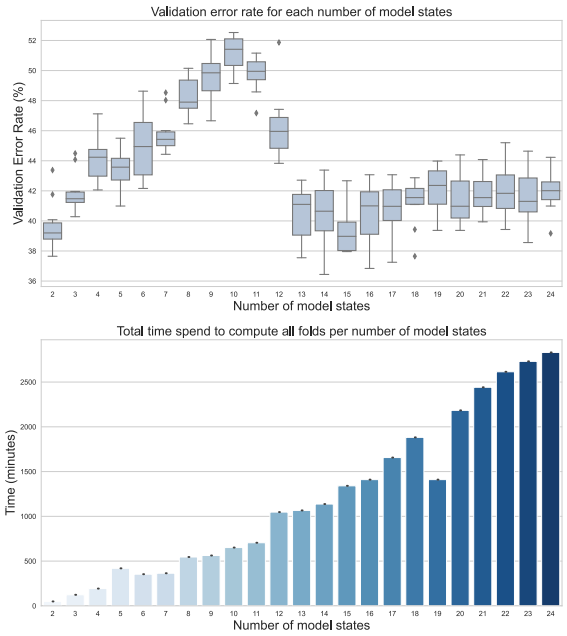


Fig. 6: Comparison of various number of given model states for the HMM model, from 2 to 12. The top panel shows a box plots of the validation error for 10-fold cross-validation of each scenario, while the bottom panel shows the total time spent computing all 10 folds in each scenario.

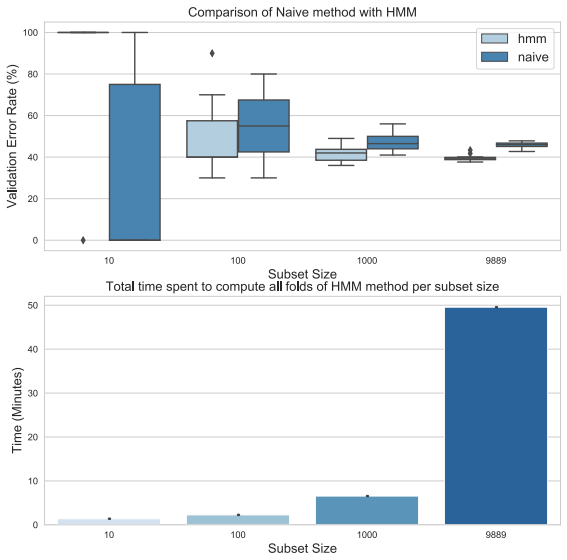


Fig. 7: A comparison of the different subset sizes on the models. The top panel shows box plots of the validation error for 10-fold cross-validation of the HMM and naive methods with 2 model states for 4 different subsets of the data. The bottom panel shows the total time taken to compute all 10 folds for the HMM method with 2 model states for 4 different subsets of the data.

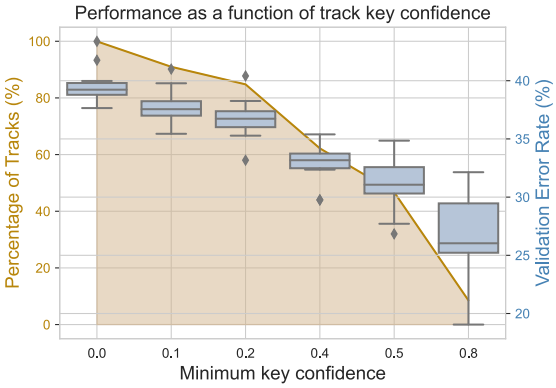


Fig. 8: The validation error rate plotted on the left y-axis and the percentage pf tracks plotted on the right y-axis of a 10-fold cross-validation of the HMM with 2 model states for an increasing key confidence minimum.

## IV. Conclusion

Attempting to automatically recognise the musical key of a track given its chroma vector using Hidden Markov Models has proven to be both more stable and accurate than the naive approach of cross-correlation. Using 2 model states resulted in a minimum computational time of about 50 minutes and a validation error rate between 38.12% and 43.38% for a 10-fold cross-validation. Additionally, the HMM method's ability to out-perform the naive method was found to increase with the size of the data-set, as well as its stability.

Although the HMM succeeds in out-performing the naive method, the validation error rate is still rather large. This could be due to a number of simplifications made to this project due to time limitations and the resources available. One of which is the assumption that the labelled mode and key of each track is correct, which could not always be the case. Further investigation of this assumption showed that the validation error decreased for tracks with higher key confidence. However, the high key confidence itself could be due to some characteristic that these tracks share that decrease the error rather than the actual confidence itself. It is therefore recommended for improvements on this task to delve deeper into what causes certain tracks to have higher confidence levels

and potentially filter based on the confidence of these key labels. In addition to this, as it is seen to improve the models with an increase in the size of the data-sets, it is expected that even larger samples would further improve these models.

Beyond what was covered in this report, there is plenty of potential for future work on the topic. One of which is the relationship between the musical genre and key and its effect on key recognition. A fundamental limitation to this project is the focus on western music and using the pitch classes associated. An interesting but challenging task worth investigating would be attempting to use similar techniques on more representative samples of music that include a variety of styles and world-wide genres. It also important to note that our method uses the given mode to estimate the overall key, a limitation that prevents key recognition without the mode which is something worth implementing in future work.

Despite the simplicity of the project, it contributes to the initial investigation of estimating musical keys with the aid of machine learning and with some further improvements could prove useful for professionals such as disk-jockeys.

## V. References


[1] Peregrine documentation [hpc documentation]. https://wiki.hpc.rug.nl/peregrine/start. (Accessed on 02/08/2021).

[2] Jeff A Bilmes et al. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute*, 4(510):126, 1998.

[3] The Editors of Encyclopaedia Britannica. Heptatonic scale. https://www.britannica.com/art/heptatonic-scale, Jul. 1998. (Accessed 8 February 2021).

[4] Ching-Wei Chen, Paul Lamere, Markus Schedl, and Hamed Zamani. Recsys challenge 2018: Automatic music playlist continuation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 527–528, New York, NY, USA, 2018. Association for Computing Machinery.

[5] Pierre A Devijver. Baum's forward-backward algorithm revisited. *Pattern Recognition Letters*, 3(6):369 – 373, 1985.

[6] T. Hastie et al. *The Element of Statistical Learning: Data Mining, Inference, and Prediction.*, volume 2. Springer-Verlag New York, 2009.

[7] Fraunhofer IIS. Fraunhofer iis index. https://web.archive.org/web/20080124200925/http://www.iis.fraunhofer.de/EN/bf/amm/projects/mp3/index.jsp, 2007. (Accessed on 02/08/2021).

[8] Carol L. Krumhansl and Edward J. Kessler. Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys. *Psychological review*, 1982.

[9] R. Mahieu. Detecting Musical Key with Supervised Learning. *Stanford University*, 2016.

[10] K. Noland and M. Sandler. Key Estimation Using a Hidden Markov Model. *Centre for Digital Music*, 2006.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[12] G. Peeters. Musical key estimation of audio signal based on hidden markov modeling of chroma vectors. 2006. (Accessed on 12/09/2020).

[13] Omologo M. Reassigned PKhadkevich, M. Reassigned spectrum-based feature extraction for GMM-based automatic chord recognition. *J AUDIO SPEECH MUSIC PROC.*, 15, 2013.

[14] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 1989.

[15] P. Skidén. Api improvements and u | spotify for developers. https://developer.spotify.com/community/news/2016/03/29/api-improvements-update/, May 2016. (Accessed on 02/09/2021).

[16] Spotify. Spotify web api. [Online; accessed December 3, 2020].


## Appendix I
## Circular shifting mathematics

For clarity, we denote vectors with an arrow symbol (i.e $\vec{v}$) and matrices with capital letters (i.e. $A$). Higher-order tensors are denoted in bold, for example as $\mathbf{X}$. Scalars are denoted with lowercase letters.

We first define a circular shifting operator $\circlearrowleft$. This operator shifts a vector by a single component in a circular fashion, for example

$$\circlearrowleft \left( \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \right) = \begin{bmatrix} d \\ a \\ b \\ c \end{bmatrix} \tag{2}$$

and similarly for a row vector

$$\circlearrowleft \left( \begin{bmatrix} a & b & c & d \end{bmatrix} \right) = \begin{bmatrix} d & a & b & c \end{bmatrix}. \tag{3}$$

We can also shift matrices, by first shifting each column and then each row:

$$\circlearrowleft \left( \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \right) \tag{4}$$

$$= \begin{bmatrix} a_{44} & a_{41} & a_{42} & a_{43} \\ a_{14} & a_{11} & a_{12} & a_{13} \\ a_{24} & a_{21} & a_{22} & a_{23} \\ a_{34} & a_{31} & a_{32} & a_{33} \end{bmatrix}. \tag{5}$$

An important property of the shift operation for matrices to note here, is that it preserves symmetry.

We denote the repeated application of the shifting operation as $\circlearrowleft^n$ for $n$ operations ($\circlearrowleft^0$ is the identity operation). We also define $\circlearrowright$ as the inverse operation of $\circlearrowleft$, and $\circlearrowleft^{-n}{=}\circlearrowright^n$.

Recall that any HMM comprises of $N$ states ($\{q_1, q_2, \ldots, q_N\} = Q$) and accepts a sequence of observations as "input". In the case of our Gaussian HMM, these observations are vectors $\vec{o_1}, \vec{o_2}, \ldots, \vec{o_T}$ in $\mathbb{R}^k$. The Gaussian HMM model consists of 4 parameters which dictate the behavior of the system, $\{A \in \mathbb{R}^{N\times N}, \vec{\pi} \in \mathbb{R}^N, \mathbf{\Sigma} \in \mathbb{R}^{N\times k\times k}, M \in \mathbb{R}^{N\times k}\}$. [2]

Here, $A$ represents the *transition probability matrix* for the model states and $\vec{\pi}$ represents the $N$ *initial state probabilities* $\pi_1, \pi_2, \ldots, \pi_N$.

$\mathbf{\Sigma}$ and $M$ represent the $N$ covariance matrices $\Sigma_1, \Sigma_2, \ldots, \Sigma_N$ and $N$ mean vectors $\mu_1, \mu_2, \ldots, \mu_N$ defining the Gaussian functions to compute the *emission probabilities* for each state given a $k$-dimensional observation. The probability of an observation $o_t$ being generated from a state $i$ is thus computed as

$$\begin{aligned} b_i(o_t) &= b(o_t, \mu_i, \Sigma_i) \\ &= \frac{1}{\sqrt{(2\pi)^k |\Sigma_i|}} e^{\left(-\frac{1}{2}(o_t-\mu_i)^T {\Sigma_i}^{-1}(o_t-\mu_i)\right)} \end{aligned} \tag{6}$$

In the training phase of our HMM, we shift each observation based on its key label to let each observation sequence represent the same key `C`. For example, we can transform a sequence of observations $O$ from the key `F#` to represent a sequence with label `C` using the shift operator:

$$\circlearrowleft^6 (O) = \{\circlearrowleft^6 (o_t) \mid o_t \in O\} \tag{7}$$

Here, 6 signifies the "difference" between the keys `C` and `F#`.

Once the model is trained, it can be used to detect sequences belonging to the `C` class since the model parameters are learned such that the likelihood score for such sequences is high. A method to classify a sequence $O$ given the model parameters into any of the 12 keys can be formalized as

$$\hat{y} = \underset{y' \in \{0,1,\ldots,11\}}{\arg\max} \; LS(\circlearrowleft^{y'} (O) \mid A, \vec{\pi}, \mathbf{\Sigma}, M) \tag{8}$$

where $LS(\cdot)$ denotes the likelihood score of the given observation sequence given the model parameters and the emission probability function as given in Equation 6. This works, but it requires us to shift each sequence 12 times. Instead, as proposed in [12], we can compute

$$\hat{y} = \underset{y' \in \{0,1,\ldots,11\}}{\arg\max} \; LS(O \mid A, \vec{\pi}, \circlearrowleft^{y'} (\mathbf{\Sigma}), \circlearrowleft^{y'} (M)). \tag{9}$$

Note that we have not yet defined $\circlearrowleft$ for a higher order tensor, and for our purposes we mean here simply $\circlearrowleft^{y'} (\mathbf{\Sigma})_{n,:,:} = \circlearrowleft^{y'} (\Sigma_n)$. Recall that the symmetry of the covariance matrices is preserved by the circular shifting operation. The method formalized in Equation 9 requires us to shift the model parameters $\mathbf{\Sigma}$ and $M$ 12 times, resulting in 12 different models. However, this is almost certainly more efficient than shifting each input observation

sequence.
Note that this procedure is executed entirely twice, once for each of the two key *modes*, major and minor.