# UNSUPERVISED LEARNING AND REFINEMENT OF RHYTHMIC PATTERNS FOR BEAT AND DOWNBEAT TRACKING

*Florian Krebs*[*]*, Filip Korzeniowski*[*]*, Maarten Grachten*[†]*, and Gerhard Widmer*[*†]

[*] Department of Computational Perception
Johannes Kepler University, Linz, Austria

[†] Austrian Research Institute for AI
Vienna, Austria

## ABSTRACT

In this paper, we propose a method of extracting rhythmic patterns from audio recordings to be used for training a probabilistic model for beat and downbeat extraction. The method comprises two stages: clustering and refinement. It is able to take advantage of any available annotations that are related to the metrical structure (e.g., beats, tempo, downbeats, dance style). Our evaluation on the Ballroom dataset showed that our unsupervised method achieves results comparable to those of a supervised model. On another dataset, the proposed method performs as well as one of two reference systems in the beat tracking task, and achieves better results in downbeat tracking.

***Index Terms***— Hidden Markov model, Viterbi training, beat tracking, downbeat tracking, clustering

## 1. INTRODUCTION

A musical work can be regarded as a composition of sounds in time. Most musical works are *metrical*, which means that the time at which sounds occur is organised into a hierarchical structure consisting of multiple pulse levels. The perceptually most prominent level is referred to as the *beat*, which coincides with the pulse most people would naturally tap their foot to. These beats are in turn organised into groups of equal numbers of beats (*measures*), with the *downbeat* denoting the first and strongest beat of each measure. The distribution of sound events within the metrical structure is referred to as *rhythm*.

Automatic analysis of the rhythmic structure of a musical work from an audio recording facilitates many aspects of music information retrieval research, such as music transcription, music similarity estimation, and music segmentation. The great challenge when designing an automatic rhythm transcription system lies in the great rhythmic variety of music. A general rhythm extraction system should work for all metres; it should handle swing, syncopations, tuplets as well as different tempo ranges. It is therefore unlikely that simple approaches such as rule-based methods can satisfy these requirements.

Whiteley *et al.* [1] proposed a system that jointly models bar position, tempo, metre, and rhythmic pattern using a hidden Markov model (HMM). Recently, we [2] have extended this model by incorporating eight rhythmic pattern states to model the eight dance styles in the data. We showed that it outperforms state-of-the-art beat and downbeat detection systems on a dataset of 697 ballroom music excerpts. However, the parameters of these rhythmic pattern states are learned in a supervised manner: the system needs a list of rhythmic pattern labels as input.

In the present paper we describe an approach to learning the model parameters (see Section 2) in an unsupervised manner. Thus, rhythmic pattern labels are no longer necessary. The proposed method is as follows: After clustering the bars of the dataset based on onset features (Section 3.1), we refine the parameter estimation using Viterbi training (Section 3.2). Finally, we apply the system to a dataset for which rhythmic pattern labels are not available (Section 4).

## 2. MODEL DESCRIPTION

In this section, we describe the beat and downbeat detection system used in this work. For a more detailed description, the interested reader is referred to [2].

We take features extracted from the audio signal as observed variables $\mathbf{y}_k$, and infer the hidden variables $\mathbf{x}_k$ of bar position, tempo, and rhythmic pattern on the basis of a hidden Markov model (HMM). Here, $k$ denotes the audio frame index with $1 \leq k \leq K$, where $K$ is the number of audio frames extracted for a piece. In an HMM, the joint probability distribution of a sequence of hidden states $\mathbf{x}_{1:K}$ and observed states $\mathbf{y}_{1:K}$ factorises as

$$P(\mathbf{x}_{1:K}, \mathbf{y}_{1:K}) = P(\mathbf{x}_1) \prod_{k=2}^{K} P(\mathbf{x}_k|\mathbf{x}_{k-1}) P(\mathbf{y}_k|\mathbf{x}_k), \quad (1)$$

where $P(\mathbf{x}_1)$ is the initial distribution of the hidden states, $P(\mathbf{x}_k|\mathbf{x}_{k-1})$ is the transition model, and $P(\mathbf{y}_k|\mathbf{x}_k)$ is the observation model. The state space of the hidden variables $\mathbf{x}_k$ is the Cartesian product of three discrete sub-state spaces:

This research is supported by the Austrian Science Fund (FWF) grant Z159 and the European Union Seventh Framework Programme through the Lrn2Cre8 project (grant agreement no. 601166).

the position inside a bar $m \in \{1, 2, ..., M\}$, the tempo $n \in \{1, 2, ..., N\}$, and the rhythmic pattern $r \in \{r_1, r_2, ..., r_R\}$.[1] Thus, a state at audio frame $k$ in this state space is written as $\mathbf{x}_k = [m_k, n_k, r_k]$.

The sequence of hidden states with the maximum a posteriori probability $\mathbf{x}^*_{1:K}$ is computed using the Viterbi algorithm as

$$\mathbf{x}^*_{1:K} = \arg\max_{\mathbf{x}_{1:K}} \{ P(\mathbf{x}_{1:K} \mid \mathbf{y}_{1:K}, \lambda) \}, \quad (2)$$

where $\lambda$ are the parameters of the model.

### 2.1. Transition model

To make inference in this large state space computationally feasible, the system is restricted to a limited number of transitions per state. We allow for three possible tempo state transitions, as modeled by the transition probabilities for $n$: if $n_k \in \{n_{min}(r_k), ..., n_{max}(r_k)\}$,

$$P(n_k|n_{k-1}) = \begin{cases} 1 - p_n, & n_k = n_{k-1}, \\ \frac{p_n}{2}, & n_k = n_{k-1} + 1, \\ \frac{p_n}{2}, & n_k = n_{k-1} - 1, \end{cases} \quad (3)$$

where $p_n$ is the probability of a tempo change, and $n_{min}(r_k)$ and $n_{max}(r_k)$ are respectively the minimum and maximum tempo corresponding to the rhythmic pattern $r_k$. Transitions to tempi outside the allowed range are assigned a zero probability.

The bar position $m_k$ at time frame $k$ is defined deterministically by the previous bar position $m_{k-1}$ and the previous tempo $n_{k-1}$.

Finally, the rhythmic pattern state is assumed to change only at bar boundaries:

$$P(r_k|r_{k-1}, m_k < m_{k-1}) = p_r(r_{k-1}, r_k). \quad (4)$$

The transition probabilities of the rhythmic pattern states $p_r(r_{k-1}, r_k)$, the tempo transition probability $p_n$, and the allowed tempo ranges $n_{min}(r_k)$ and $n_{max}(r_k)$ are learned from data as described in Section 3.

### 2.2. Observation model

The observed variables $\mathbf{y_k}$ are computed using the *LogFiltSpecFlux* method introduced in [3], calculated for two frequency bands (below 250 Hz and above 250 Hz). The observation likelihood $P(\mathbf{y}_k|m_k, r_k)$ is modeled by a Gaussian mixture model (GMM) with two components. In order to obtain a manageable number of parameters to learn, the observation probabilities were assumed to be constant for the duration of a 64th note.

[1] In this paper, we used $M = 1216$, $N = 23$.

## 3. LEARNING

The model's rigid structure reduces the set of state transition parameters to $n_{min}(r_k)$, $n_{max}(r_k)$, $p_n$, and $p_r(r_{k-1}, r_k)$. Learning thus mainly focuses on extracting rhythmic patterns in form of the observation likelihood $P(\mathbf{y}_k \mid m_k, r_k)$ from the data. We obtain all parameters in a two-phase process: First, a simple k-means approach gives us an initial model; then, we refine this model via multiple runs of the segmental k-means algorithm [4], also called *Viterbi training*.

We chose the parameter estimation methods based on the available data (see Section 4.1 for details). If we had fully annotated data samples, we could simply determine all transition probabilities by counting, and learn the observation model by a single maximum likelihood estimation. However, our training data is only partially labelled: We have annotated beat and downbeat times, but no information on the bar positions between those annotations[2]. Additionally, manual beat annotations tend to be imprecise, as mentioned in [5]. Therefore, we can determine neither rhythmic patterns nor transitions directly.

We briefly describe the initialisation process in Section 3.1 before detailing the model refinement in Section 3.2.

### 3.1. K-Means Initialisation

Algorithms for learning the parameters of a HMM usually modify a given initial model towards a specified goal by minimising an objective function. While a random initial model will suffice in some cases, an informed initialisation often provides the optimiser with a better starting point, resulting in faster convergence. In our system, initial values for tempo transitions are set by hand ($p_n = 0.01$). Parameters concerning rhythmic patterns, however, are determined from training data.

Rhythmic patterns define the observation likelihood $P(\mathbf{y}_k \mid m_k, r_k)$ as described in Section 2.2. Given many degrees of freedom, finding sensible initial values is crucial. To compute these, we apply the following steps:

1. Calculate the frequency split *LogFiltSpecFlux* for each audio file in the training corpus.
2. Split the feature values into single bars based on the ground truth annotation.
3. Group the features for each bar into 64 bins equally spaced in time within each bar. This results in 128 values per bar, since we compute the onset feature for two distinct frequency bands. Determined by the ground truth, for metres other than 4/4, we use accordingly fewer bins (e.g., 48 bins for a 3/4 metre).
4. In addition to these 'bar-level' patterns, compute the average of all bars that belong to a song to obtain a 'song-

[2] We might know that audio frames $k$ and $m$ are at the first and second beat of a bar, but do not know which bar position exactly an audio frame $l$, $k < l < m$, has.

level' pattern. These will be beneficial for music with constant rhythm like Ballroom music.

5. Divide bar- and song-level patterns into $R$ clusters using the k-means method.
6. Using the cluster labels for each instance, learn the parameters of $P(\mathbf{y}_k|m_k, r_k)$, $p_r(r_{k-1}, r_k)$, $n_{min}(r_k)$ and $n_{max}(r_k)$, representing the rhythmic patterns, their interaction and the tempo range of a pattern respectively.

The resulting patterns constitute the initial model that is fed into the model refinement or learning process.

### 3.2. Model Refinement

The semantics of the hidden state sequence for a data sample determine its segmentation. We thus aim to increase the probability of obtaining the correct state sequence when decoding the given observation sequence. To this end, we apply Viterbi training [4], which adapts the model's parameters such that the probability of the decoded state sequence increases. To ensure that the decoded state sequence corresponds to the desired (correct) segmentation result, we apply methods for *Partially-Hidden Markov Models* [6], which allow us to narrow down possible decoding paths to a specified window around the annotated beat structure.

#### *3.2.1. Viterbi Training*

Viterbi training works similarly to the well-known Baum-Welch method, with minor but critical alterations. It was first described under the name *segmental k-means algorithm* in [4]. The learning process repeats the following two steps until convergence: 1) decoding using the Viterbi algorithm and 2) parameter re-estimation. In doing so, at each step it selects new model parameters $\bar{\lambda}$ such that

$$\bar{\lambda} = \arg\max_{\lambda} \{ P(\mathbf{x}^*_{1:K} \mid \mathbf{y}_{1:K}, \lambda) \}, \tag{5}$$

where $\lambda$ are the current model's parameters, and $\mathbf{x}^*_{1:K}$ is the decoded state sequence given by the Viterbi algorithm as defined in Eq. 2.

After decoding, the algorithm re-estimates the parameters in a straightforward way: $p_n$ and $p_r(r_{k-1}, r_k)$ are computed by counting the corresponding occurrences in the decoded paths; similarly, we find $p_n$, the parameter defining the probability of a tempo change in tempo, and $n_{min}(r_k)$ and $n_{max}(r_k)$, which define the possible tempo range for a given pattern. Rhythmic patterns are created using the same method as in the initialisation, with the respective $r_k$ instead of the clustering results as labels. The re-estimated parameters maximise the probability of the decoded state sequence.

Compared to the Baum-Welch algorithm, Viterbi training has some favourable properties for our use case. First and foremost, it uses a different objective function, as Eq. 5 shows. Instead of maximising the likelihood of the data (maximum likelihood estimation, MLE), it maximises the a-posteriori probability (MAP) of the most probable state sequence. MLE is more closely related to a classification scenario - improvements in segmentation quality come incidentally, but are not the target of the optimisation. MAP maximisation, however, is related to the very problem we address in this paper - we use it to increase the probability of obtaining a state sequence that corresponds to the correct segmentation. The results in [7] also show that better segmentation results can be obtained using Viterbi training.

#### *3.2.2. Decoding in Partially-Hidden Markov Models*

The standard formulations of the common inference algorithms for HMMs are based upon the assumption that only the observed variables are known. Parameter learning methods such as the Baum-Welch algorithm do not require – and cannot incorporate – knowledge about the hidden variables of a model. In [6], these algorithms are modified such that uncertain and imprecise labels, in our case beat annotations, can be utilised for inference and to support learning of HMMs. Here, we use *evidential Viterbi decoding* in the model refinement step of our training method. Due to space limitations, we can only briefly describe the difference from regular Viterbi decoding. For an in-depth description, we refer to [6].

At each new time step $k$, the Viterbi algorithm computes *Viterbi messages* $\delta_k(s)$, which represent the probability of the most probable state sequence that ends at state $s$. Based on our belief of the true bar position at this time, we can also define the *plausibility* $pl_k(s)$ of being at state $s$ at time step $k$. As derived in [6], the Viterbi messages in the evidential Viterbi algorithm are

$$\delta'_k(s) = \delta_k(s) \cdot pl_k(s),$$

while the remaining steps of the algorithm are unmodified.

We define the plausibilities based on the ground truth annotations. This results in uniform plausibilities at time steps between beats, causing the algorithm to behave like the standard decoding method. The behaviour changes within a range around the beat annotations: We force the decoded path to hit the annotated beat within $\pm$ 17.5% of the current inter-annotation interval around it. To this end, we assign a plausibility of zero to states corresponding to positions outside of this range, while keeping it uniform for those within. The allowed range corresponds to the tolerance window of the CMLt evaluation metric (see Section 4.2).

Thus we can alleviate the impact of imprecise annotations while forcing the algorithm to follow the correct segmentation path within the allowed tolerance window.

## 4. EXPERIMENTS

In this section, we describe the experimental setup to show the effect of the proposed parameter learning scheme on beat and downbeat tracking accuracy. The experimental results are obtained using a 10-fold cross-validation, where each fold was designed to have the same distribution of dance styles or metre. We used ten iterations of Viterbi training in the refinement stage. We employed a multiple comparison procedure [8] to determine whether the differences between results are statistically significant with an alpha level of 0.05. In the training stage, we used 'song-level' patterns for the Ballroom dataset, and 'bar-level' patterns for the Hainsworth dataset, which are described in the following.

### 4.1. Datasets

We evaluated the proposed learning scheme on two datasets which provide beat and downbeat annotations.

The **Ballroom** dataset comprises ballroom dance music and is expected to consist of clear and constant rhythmic patterns, which makes it suitable for pattern extraction and modeling tasks. It was originally assembled for the tempo induction contest organised during the International Conference on Music Information Retrieval (ISMIR 2004) [9]. The dataset consists of 697 excerpts of music, each 30 seconds long. [3]

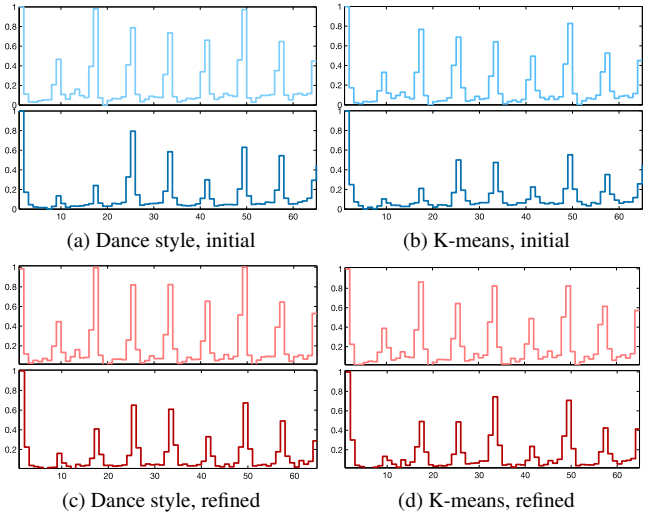
In order to investigate whether the proposed learning scheme also works for music with less constant rhythmic patterns, we chose to use the **Hainsworth** dataset, which has also been used by several other authors, e.g. [10, 11]. It consists of 222 audio files of various genres, each between 30 and 60 seconds long.

### 4.2. Evaluation metrics

A variety of measures for quantifying the performance of beat tracking systems exist. See [12] for a detailed overview. We give results for the following three metrics:

**CMLt** is a continuity-based metric that describes the percentage of beats that belong to a continuous group. For a beat to belong to a continuous group, the previous and the current beat have to be within the tolerance window of $\pm$ 17.5% of the current inter-annotation interval around the annotations. Only beats at the correct metrical level are taken into account. **AMLt** is computed the same way as CMLt, but beats are also allowed to occur at half or double the correct tempo, as well as on the offbeat. **Db-Fmeas** is the generic F-measure often used in the field of information retrieval applied to the downbeat detection task. A detected downbeat is considered correct if it falls within a $\pm$ 70ms window around an annotated downbeat.

[3] Annotations available at https://github.com/CPJKU/BallroomAnnotations (v1.1)



(a) Dance style, initial (b) K-means, initial

(c) Dance style, refined (d) K-means, refined

**Fig. 1**. Two learned rhythmic patterns, before (Figs. (a), (b)) and after refinement (Figs. (c), (d)). The x-axis defines the position within a bar in a 64$^{\text{th}}$ note grid, the y-axis represents the normalised mean onset values for a position. Each pattern consists of two frequency bands: the lower, darker plots represent the low band, the lighter, upper plots the high band.

| System | CMLt | AMLt | Db-Fmeas |
|---|---|---|---|
| PS8_dancestyles [2] | 75.4 | 88.3 | 69.7 |
| PS8_dancestyles (refined) | 78.8 | 89.1 | 70.0 |
| PS8_kmeans_b | 74.3 | 87.5 | 63.3 |
| PS8_kmeans_b (refined) | 78.4 | 88.5 | 70.8 |
| Ircambeat [11] | 57.1 | 84.1 | 42.1 |
| Böck [13] | 81.1 | 84.8 | - |

**Table 1**. Beat and downbeat tracking accuracy on the *Ballroom* dataset.

## 5. RESULTS AND DISCUSSION

In Tables 1 and 2 we show results for the following systems: For the Ballroom dataset, **PS8_dancestyles**, the model proposed in [2], where dance style labels are used to learn the rhythmic patterns[4]; **PS8_kmeans_b**, which uses our proposed pattern learning method with $K$=8 patterns trained on the Ballroom dataset; For the Hainsworth dataset, **PS3_metre**, which uses $K$=3 patterns, where each corresponds to a musical metre present in the dataset (2/4, 3/4 and 4/4); **PS8_kmeans_h**, the same as PS8_kmeans_b, but trained on the Hainsworth dataset. Systems that use the proposed refinement stage (Section 3.2) are marked by the word **refined** in brackets. Additionally, we show results for **Ircambeat** [11] and **Böck** [13] as reference. We obtained beat detections for the reference systems directly from the respective authors.

As can be seen from Table 1, the proposed PS8_kmeans_b

[4] Results may vary from those published earlier because of different training and evaluation strategies.

| System | CMLt | AMLt | Db-Fmeas |
|---|---|---|---|
| PS3_metre | 49.3 | 72.9 | 34.4 |
| PS3_metre (refined) | 52.1 | 73.7 | 36.8 |
| PS8_kmeans_h | 67.4 | 81.8 | 43.0 |
| PS8_kmeans_h (refined) | 66.7 | 82.6 | 45.8 |
| Ircambeat [11] | 60.8 | 81.1 | 40.7 |
| Böck [13] | **77.2** | 82.1 | - |

**Table 2**. Beat and downbeat tracking accuracy on the *Hainsworth* dataset.

(refined) method performs equally well as the PS8_dancestyles model in all metrics. Visual inspection of the learned rhythmic patterns revealed that the unsupervised process converged to patterns similar to those produced by supervised learning, which uses the dance-style labels for initialisation (compare left and right columns of Fig. 1, for example). Further, the four variants of PS8 outperformed the reference systems, with the exception of Böck for the CMLt metric.

Our learning method enables modelling multiple rhythmic patterns on datasets for which rhythmic pattern labels are not available. Table 2 shows results on such a dataset. Again, it seems favorable to use more than one pattern per metre, as shown by the superior performance of the PS8 variants in comparison to the PS3_metre models. Compared to the other four reference systems, we do not find any statistically significant difference between our system and the reference systems (except Böck for the CMLt metric).

The superior performance of Böck in the CMLt metric might be due to the fact that it was trained on a larger dataset than the PS8_kmeans_h model, which was trained only on the Hainsworth dataset. Future work should therefore concentrate on learning rhythmic patterns on a much larger dataset.

Finally, it is unclear under which conditions the proposed refinement increases the performance of a model. While we found a significant improvement in the CMLt and the Db-Fmeas metrics when using Viterbi refinement on the Ballroom dataset, results are less clear-cut on the Hainsworth dataset: Applying the refinement method, we observed an improvement in the downbeat tracking performance, but also a slight decrease in beat tracking performance, although neither increase nor decrease were statistically significant. One possible explanation might be that the (arbitrarily) chosen number of rhythmic patterns did not match the content of the dataset.

## 6. CONCLUSION

In this paper, we have proposed a method of learning the parameters of the rhythm description model described in [2]. The new method learns suitable patterns in an unsupervised way, given only downbeat and beat locations. We have shown that the proposed unsupervised method achieves results comparable to those of the supervised model on the Ballroom dataset. On the Hainsworth dataset, the proposed method performs as well as one of two reference systems, and is only outperformed by Böck's beat tracker in the CMLt metric. Future work will concentrate on learning rhythmic patterns from a larger dataset and will investigate how the optimal number of rhythmic patterns can be found for a given dataset.

## REFERENCES


[1] N. Whiteley, A. Cemgil, and S. Godsill, "Bayesian modelling of temporal structure in musical audio," in *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR)*, Victoria, 2006.

[2] F. Krebs, S. Böck, and G. Widmer, "Rhythmic pattern modeling for beat and downbeat tracking in musical audio," in *Proceedings of the 14th International Conference on Music Information Retrieval (ISMIR)*, Curitiba, 2013.

[3] S. Bock, F. Krebs, and M. Schedl, "Evaluating the online capabilities of onset detection methods," in *Proceedings of the 13th International Conference on Music Information Retrieval (ISMIR)*, Porto, 2012.

[4] Lawrence R. Rabiner, Jay G. Wilpon, and Bling-Hwang Juang, "A segmental k-means training procedure for connected word recognition," *AT&T Technical Journal*, vol. 65, no. 3, pp. 21–31, 1986.

[5] Simon Dixon, Werner Goebl, and Emilios Cambouropoulos, "Perceptual smoothness of tempo in expressively performed music," *Music Perception*, vol. 23, no. 3, pp. 195–214, Feb. 2006.

[6] E. Ramasso, "Contribution of belief functions to hidden markov models with an application to fault diagnosis," in *IEEE International Workshop on Machine Learning for Signal Processing, 2009. MLSP 2009*, Sept. 2009, pp. 1–6.

[7] Luis Javier Rodríguez and Inés Torres, "Comparative study of the baum-welch and viterbi training algorithms applied to read and spontaneous speech recognition," in *Pattern Recognition and Image Analysis*, p. 847–857. Springer, 2003.

[8] Y. Hochberg and A. Tamhane, *Multiple comparison procedures*, John Wiley & Sons, Inc., 1987.

[9] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, "An experimental comparison of audio tempo induction algorithms," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1832–1844, 2006.

[10] S. Hainsworth and M. Macleod, "Particle filtering applied to musical tempo tracking," *EURASIP Journal on Applied Signal Processing*, vol. 2004, pp. 2385–2395, 2004.

[11] G. Peeters and H. Papadopoulos, "Simultaneous beat and downbeat-tracking using a probabilistic framework: theory and large-scale evaluation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 6, pp. 1754–1769, 2011.

[12] M. Davies, N. Degara, and M.D. Plumbley, "Evaluation methods for musical audio beat tracking algorithms," *Queen Mary University of London, Tech. Rep. C4DM-09-06*, 2009.

[13] S. Böck and M. Schedl, "Enhanced beat tracking with context-aware neural networks," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2011.