Weakly-supervised segmentation applied to audio-to-score alignment

Émile Enguehard

14th September 2016

Contents

1 Introduction: audio-to-score alignment

Given a music score and a recording of the same piece, a common problem consists in trying to find a mapping between the score's beats and the recording's timestamps. This question is known as *audio-to-score alignment*. The difficulty of it lies in the fact that even classical pieces with detailed scores are always performed in slightly different manners; this is due to the performers' effects as well as sometimes their mistakes. Thus, two perfomances of the same piece may have very different local tempi and temporal deviations from one another.

In addition to its inherent applicative potential (for instance as a teaching aid in music schools), audio-to-score alignment can be used as a frontend for a number of common problems in music processing, including query-by-humming, where we seek to identify a piece hummed or played by a human [1], automatic transcription [2], audio editing [3], automatic page turning (useful for musical performers) [4], automatic accompaniment [5] etc. It can also help generating large datasets of annotated recordings, for use in any machine learning questions applied to music.

Various approaches have been applied to score alignment; they tend to fall in two broad categories. First, penalization-based methods try to minimize an arbitrary function of the data; they often use the Dynamic Time Warping algorithm (DTW) (for instance [6]). The second category includes methods based on probabilistic models of the data, such as Hidden Markov Models (HMMs) or more elaborate graphical models (for instance [7]).[1] In both cases, those methods usually seek to minimize a pre-determined similarity measure between the score and the audio, such as log-likelihood in a graphical model with pre-trained parameters. Some works have also attempted to learn appropriate features [8], using fully annotated samples where every score event is associated to a timestamp. Those samples tend to be small and in short supply, as in the absence of good enough audio-to-score alignment algorithms, recordings have to be annotated by hand, a process that is both cumbersome and prone to error.

The approach described in this work, proposed by [9], circumvents that problem by only requiring pairs of a score and a recording to be trained, which are available in much greater numbers; for this reason we consider it to be *weakly-supervised*. In effect, we optimize the problem for both the alignment and the data representation at the same time, using convex optimization methods. The choice of cost function is inspired by *discriminative clustering* [10, 11]. Discriminative methods have received some attention recently in many domains as an alternative to generative models that often leads to more easily tractable optimization problems. It should be noted that audio-to-score alignment can be done in real time (a.k.a. *online*) or *offline*, using the whole signal at once; our method only applies readily to the offline problem.

The focus of this work is to develop and improve on the alignment method proposed by [9]. In section 2, we start by describing their approach to alignment; section 3 contains our proposed improvements on the model of [9], and section 4 describes the experimental results we obtained. Our contributions consist in: (i) proposing a way to handle polyphonic data, (ii) designing a more efficient rounding procedure for the convex relaxation of [9], based on an linear-time version of the HSMM Viterbi algorithm, (iii) investigating two alternative, more versatile optimization strategies also based on that algorithm, one through a different convex relaxation, the other through alternating optimization, and (iv) a more thorough experimental comparison of our approach with other techniques, including a generative method, using real-world data.

[1]Both HMMs and the DTW algorithm will receive further attention in this report.

2 Weakly-supervised segmentation

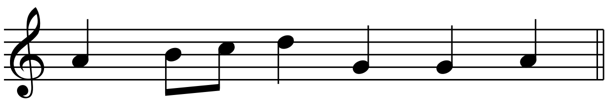
2.1 Description of the problem

The problem we seek to solve is that of *alignment*, specifically audio-to-score alignment. Our data consists in two time series. First, we have prior information: the *template* (in our case, a music score) indicates an ordered sequence of events belonging to K different classes (music notes, chords or rests). As posterior information, we consider a d-dimensional *signal* (here, sound spectrograms).[2]

This can be written in matrix form:

- The template is a matrix Φ with dimensions $E \times K$, where E is the number of events. $\Phi_{i,k}$ is 1 if event i belongs to class k, 0 otherwise.
- The signal is a matrix X with dimension $T \times d$, where T is the "duration" of the signal.

Consider, for instance, the following piece:



This piece has $E = 7$ events belonging to $K = 5$ classes. If we rank the notes in order from middle G to high D (so that middle G is 0, middle A is 1, and so on), we can represent it by the matrix in figure 1.

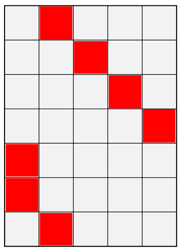


Figure 1: An example of template matrix Φ.

Our goal is to recover correspondences between the signal and the events, that is, find which event $i \in \{0, \dots, E-1\}$ is happening at time $t \in \{0, \dots, T-1\}$. We have the following constraints: only one event is happening at any given time step (no overlaps), and exactly one (no jumps), and the events follow one another in the sequence $0, 1, \dots, E-1$, with event 0 starting at time 0 and event $E-1$ ending at time T.

On simple way to represent an alignment is to use the *onset times* of the events, that is, the sequence $t_0, \dots, t_{E-1}$ such that event i starts at time t_i. Equivalently, we may use the duration l_i of each event. A third way introduces an "alignment path" $y(t) \in \{0, \dots, E-1\}$, with the following constraints: $y(0) = 0$, $y(T-1) = E-1$, and $y(t) \leq y(t+1) \leq y(t) + 1$.

The authors of [9] choose to represent the alignment path in matrix form, with an alignment matrix $Y \in \mathbb{R}^{T \times E}$ where $Y_{t,i}$ is 1 if at timestep t, event i is occurring, i.e. if $y(t) = i$, and 0, otherwise. Thus Y follows the following constraints: $Y_{0,0} = 1$, $Y_{T-1,E-1} = 1$, and if $Y_{t,i} = 1$ then only one of $Y_{t+1,i}$ and $Y_{t+1,i+1}$ is 1. They denote the set of valid Y matrices as $\mathcal{Y}$, and its convex hull as $\overline{\mathcal{Y}}$.

To Y they associate an *assignment matrix*, Z, with dimensions $T \times K$, indicating to which class the event at each timestep belongs. Note that Z is less informative than Y since we

[2]Thus, our approach could also be applied to alignment of speech to text, for instance.

can't separate consecutive occurrences of the same event. Y and Z are related by the following equation:

$$Z = Y\Phi. \tag{1}$$

Figure 2 further illustrates this by showing what an alignment on the score above could look like, and the associated Z matrix. Note that the template, as defined above, only includes information about the sequence of events. In an actual music score, we also have information about the relative length of each event. For instance, the 7 events of our score have relative lengths $(1, \frac{1}{2}, \frac{1}{2}, 1, 1, 1, 1)$; the alignment proposed in figure 2 takes $T = 12$ and follows those relative lengths exactly.

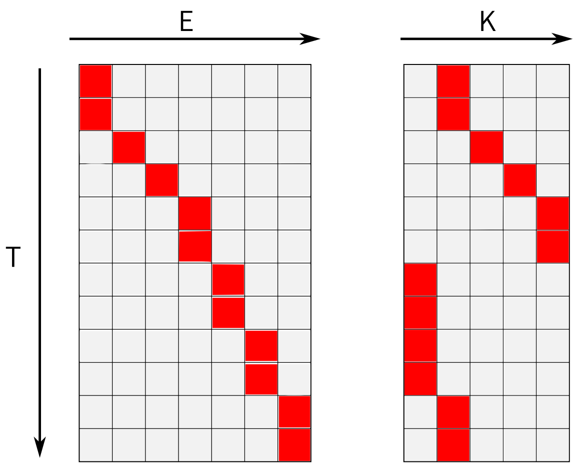


Figure 2: The relationship between Y (left) and Z (right). We take $T = 12$ and follow the score exactly.

In practice we consider several template-signal pairs. This doesn't change anything to the following discussions, as we can simply stack our various Z and X matrices. In the following discussions, we will denote as n the number of template-signal pairs that we use, as Y_j, Z_j, Φ_j and X_j the alignment matrix, assignment matrix, template and observed features associated with each pair, as X and Z the stacked data and assignment matrices and as Y and Φ the sequence of alignment and template matrices when it makes sense.

Our goal is to recover, for each signal, the "true" alignment matrix Y_{gt} (for "ground truth").

2.2 Mathematical formulation

2.2.1 Discriminative clustering

To that effect, we will adopt an approach inspired by discriminative clustering. Discriminative models are probabilistic models often contrasted with generative models. [11, 10] A *generative* model is one where the relationship between the observed features X and the latent labels Z is based on the following graphical model:

$$Z \longrightarrow X \tag{2}$$

By constrast, a *discriminative* approach will use the following graphical model:

$$X \longrightarrow Z \tag{3}$$

In the context of supervised classification, an example of a generative method would be the use of Gaussian Mixture Models (GMM), as well as the confusingly-named Linear Discriminant

Analysis (LDA). Examples of discriminative methods include linear or logistic regression and neural networks.

Discriminative models in unsupervised learning have gained traction more recently than generative ones. They have been shown on occasion to have better predictive performances than generative ones, and to yield more easily tractable or adaptable cost functions, as well as being less dependent on affine transformation of the data. [10, 12]

An example of a generative approach to alignment is the use of Hidden Semi-Markov Models (HSMMs), which we will come back to in section 3.2.1.

2.2.2 Application to alignment

Our approach, after [9], to alignment is inspired by discriminative clustering: we will learn K affine classifiers, represented by the pair (W, b) below, of the data X jointly with alignments Y. More precisely, we will solve the following problem:

$$\begin{aligned} &\text{minimize } \frac{1}{n}\sum_{j=1}^{n} \|Y_j\Phi_j - X_jW - \mathbf{1}b^T\|_F^2 + \lambda\|W\|_F^2 \\ &\text{s.t. } Y_j \in \mathcal{Y}_j,\ W \in \mathbb{R}^{d\times K},\ b \in \mathbb{R}^K \end{aligned} \tag{4}$$

The second term is a Tikhonov regularization. $\|\cdot\|_F$ denotes the Frobenius norm: $\|A\|_F^2 = \text{tr}(A^TA)$. The vector b stands for “bias” and serves to re-center the data. The $\frac{1}{n}$ factor reduces the dependency on the size of the dataset. A slight variant of this cost function is used for clustering by [10].

We consider this approach to be *weakly-supervised*, as it doesn't rely on prior information on the categories or the data beyond the redudancy information brought by the knowledge of Φ. Specifically, the classifers W and biases b are learnt as the same time as the alignment Y, and the learning process doesn't involve annotated data (in the form of known signal/alignment pairs).

This is an unconstrained minimization problem in W and b, and we can derive closed-form expressions for their optimal values W^* and b^*:

$$W^* = \Gamma Z, \tag{5}$$

$$b^* = \frac{1}{T}(Z - XW^*)^T\mathbf{1}_T, \tag{6}$$

where:

$$\Gamma = (X^T\Pi_T X + n\lambda I_d)^{-1}X^T\Pi_T, \tag{7}$$

$$\Pi_T = I_T - \frac{1}{N}\mathbf{1}_T\mathbf{1}_T^T, \tag{8}$$

N being the total number of data points. Π_T is a projection matrix: multiplying by Π_T on the left side is equivalent to substracting the mean from each column.

The resulting objective function is:

$$F(Z) = \text{tr}(AZZ^T), \tag{9}$$

where:

$$\begin{aligned} A &= \frac{1}{n}\Pi_T\left(I_T - X(n\lambda I_d + X^T\Pi_T X)^{-1}X^T\right)\Pi_T \\ &= \frac{1}{n}\left(\Pi_T - \Pi_T X\Gamma\right). \end{aligned} \tag{10}$$

We also have the gradients:

$$\nabla_Z F(Z) = 2AZ \tag{11}$$

$$\nabla_{Y_j} F(Y) = \left(\nabla_{Z_j} F(Z)\right)\Phi_j^T. \tag{12}$$

2.2.3 Optimization techniques

Both discriminative and generative models often come down to intractable or difficult optimization problems. In our case, we have to minimize a quadratic function of Y over the discrete set $\mathcal{Y}$. A common strategy (used by [10] and [11]) consists in expressing the objective function in terms of the *equivalence matrix* M or the *normalized equivalence matrix* $\tilde{M}$. If $Z \in \mathbb{R}^{N\times K}$, where N is the number of data points and K is the number of classes, is an assignment matrix, then:

$$M = ZZ^T \qquad \qquad \tilde{M} = Z(Z^TZ)^{-1}Z^T \tag{13}$$

The properties of those matrices enable the authors of [10] and [11] to express the cost as a linear function; then, they use convex relaxation of the set where those matrices evolve to solve the LP. The issue with this approach is that those two matrices evolve in a set whose convex hull cannot be expressed in a simple way; thus approximative convex relaxations have to be used instead. In [9], this issue is avoided by taking a convex relaxation in Y instead.

2.3 Prior information

2.3.1 Motivation

In discriminative clustering, the solution that assigns all data points to the same cluster is often trivially optimal; more generally, degenerate solutions with overly unbalanced clusters are a frequent risk. The opposite problem is found in generative clustering, where degenerate solutions assign each point to its own cluster.

Various heuristical strategies are used to counter this: hard limits on the number of classes (this doesn't readily apply to our problem where the number of segments is fixed), minimal cluster sizes, [10] or adding smooth regularization to the objective; for instance, [11] introduces the entropy of cluster sizes as a penalization:

$$h(Y) = -H\left(\frac{1}{\mathbf{1}^TY\mathbf{1}}Y^T\mathbf{1}\right) \tag{14}$$

where H is the usual entropy function: $H(x) = -\sum x_i \log x_i$. This tends to promote equal cluster sizes.

Following [9], we will adopt a similar strategy, using the information on events' duration found in the music score. Indeed, as we already noted, the music score provides additional information rather than just the template Φ; it also defines relative lengths for all events, which induces an "expected" alignment $\overline{Y}$, the alignment that follows the score exactly.

2.3.2 Global prior

We considered two different ways of adding this information to our data: the first one is what [9] calls the "global" prior, which penalizes the mean differences between onsets of events in the computed and expected alignments, by adding a term:

$$h_{\text{global}}(Y) = \frac{1}{n}\sum_{j=1}^{n} \|(Y_j - \overline{Y_j})L_j^T\|_F^2, \tag{15}$$

where L_j is a lower triangular matrix with only ones. This is equivalent to the "area loss" between the two alignment matrices. It can also be interpreted as the cumulated delay between the two alignments; denoting as $(t_0, \ldots, t_{E-1})$ the onset times of events following alignment Y, and $(t'_0, \ldots, t'_{E-1})$ those based on alignment Y', we have:

$$\|(Y - Y')L^T\|_F^2 = \sum_{i=0}^{E-1} |t_i - t'_i|. \tag{16}$$

h_{global} reduces to an affine function on $\mathcal{Y}$:

$$h_{\text{global}}(Y) = \frac{1}{n}\sum_{j=1}^{n}\left(\operatorname{tr}\left(Y_j^T(\operatorname{diag}(L_j^T L_j)\mathbf{1}_{E_j}^T - 2\overline{Y}_j L_j^T L_j)\right) + \|\overline{Y}_j L_j^T\|_F^2\right). \tag{17}$$

This is no longer true on $\overline{\mathcal{Y}}$ since it results from the following non-linear relation:

$$\forall Y \in \mathcal{Y},\; Y^T Y = \operatorname{diag}(Y^T\mathbf{1}_T) \tag{18}$$

where the diag operator maps a vector in $\mathbb{R}^p$ to a diagonal matrix in $\mathbb{R}^{p\times p}$.

2.3.3 Local prior

[9] also introduces a "local prior" inspired by Hidden Semi-Markov Models (HSMMs, cf section 3.2.1). To each event i we associate an expected duration $l_i^{(j)}$ and add the following term:

$$h_{\text{local}}(Y) = -\frac{1}{n}\sum_{j=1}^{n}\sum_{i=0}^{E_j-1}\log P(t_{i+1}^{(j)} - t_i^{(j)}; l_i^{(j)}). \tag{19}$$

where P is some probability distribution on positive reals or integers with a single real parameter. We call this regularization "local" because it penalizes each cluster size independently.

One choice we considered for P is that of (truncated) homoscedatic Gaussians (with mean l and fixed variance), resulting in:

$$h_{\text{local}}(Y) = \frac{1}{n}\sum_{j=1}^{n}\sum_{i=0}^{E_j-1}(t_{i+1}^{(j)} - t_i^{(j)} - l_i^{(j)})^2 = \frac{1}{n}\sum_{j=1}^{n}\|(Y_j - \overline{Y}_j)^T\mathbf{1}_{T_j}\|_2^2. \tag{20}$$

This has the advantage to induce a quadratic cost; thus, our optimization problem stays quadratic. It is used in the experiments of [9]. Another possibility that also lets us stay in a QP setting is that of heteroscedatic Gaussians, with variance proportional to l:

$$h_{\text{local}}(Y) = \frac{1}{n}\sum_{j=1}^{n}\sum_{i=0}^{E_j-1}\left(\frac{t_{i+1}^{(j)} - t_i^{(j)} - l_i^{(j)}}{l_i^{(j)}}\right)^2 = \frac{1}{n}\sum_{j=1}^{n}\|(\overline{Y}_j^T\overline{Y}_j)^{-1}(Y_j - \overline{Y}_j)^T\mathbf{1}_{T_j}\|_2^2. \tag{21}$$

This latter cost function is consistent with a number of properties of musical data; in particular, the sub-path of minimum cost is always the one that follows exactly the score, even if that's in a different tempo from the global one. Log-concavity of the distribution is a necessary condition for this to hold.

2.4 Optimization algorithm

We cannot solve our optimization problem as is; instead we follow [9] in considering its convex relaxation:

$$\begin{aligned} &\text{minimize } F(Y) + \mu\, h_{\text{global}}(Y) + \nu\, h_{\text{local}}(Y) \\ &\text{s.t. } Y \in \overline{\mathcal{Y}} \end{aligned} \tag{22}$$

μ and ν are hyperparameters that have to be set by the experimenter.

We denote the full objective function as $G(Y)$. Note that G is still a quadratic function.

2.4.1 Dynamic Time Warping

We can solve linear problems on $\mathcal{Y}_j$ quickly (in time $O(TE)$) thanks to the Dynamic Time Warping (DTW) algorithm. The DTW algorithm, often used in alignment problems, can minimize objectives of the form $\mathrm{tr}(Y_j^T C)$, where C is a cost matrix; $C_{t,i}$ is the cost of being in event i at time t. This is done by recursively computing the minimal cost $D_{t,i}$ to be in event i at time t, with the following formula:

$$D_{0,0} = C_{0,0} \tag{23}$$

$$D_{t,i} = +\infty \qquad \text{if } t < i \tag{24}$$

$$D_{t+1,i+1} = \min\left\{D_{t-1,i}, D_{t-1,i-1}\right\} + C_{t,i} \tag{25}$$

We keep track of previous results to reduce computation time, hence the "dynamic" qualifier. At the end, the optimal alignment is the one that yielded the cost $D_{T-1,E-1}$.

2.4.2 Frank-Wolfe method

Since $\overline{\mathcal{Y}}_j$ is a convex polytope, solving linear problems on $\overline{\mathcal{Y}}_j$ is the same as solving them on its set of vertices $\mathcal{Y}$. This means we can also solve linear problems on $\overline{\mathcal{Y}}_j$ using DTW. This remark prompts the use of a Frank-Wolfe algorithm (FW, a.k.a. Conditional Gradient algorithm, CG): in this optimization method, search directions are found by minimizing a local linear approximation of the objective function. Thus, in our case, at every iteration k, the new search direction is found by calculating:

$$Y_j'^{(k)} \in \underset{Y_j \in \mathcal{Y}_j}{\arg\min}\ \mathrm{tr}\left(Y_j^T \nabla_{Y_j} G(Y^{(k)})\right). \tag{26}$$

We can then pick a new point using a step size γ_k:

$$Y_j^{(k+1)} = \gamma_k Y_j'^{(k)} + (1-\gamma_k) Y_j^{(k)}. \tag{27}$$

A possibility is to use the "universal" step $\gamma_k = \frac{2}{k+1}$. In our case, since the objective is a quadratic function of Y, there is a closed-form expression of the optimal step.

In both cases, the algorithm can be shown to converge linearly towards the optimum. [13]

2.4.3 Rounding

The algorithm yields a point Y^* in $\overline{\mathcal{Y}}$ from which we may wish to recover a proper alignment in $\mathcal{Y}$. This procedure can be referred to as *rounding*.

Various rounding schemes can be realised in time $O(TE)$ using the DTW algorithm:

1. Minimize $\|Y_j - Y_j^*\|_F^2$; this is equivalent to maximizing $\mathrm{tr}(Y_j^T Y_j^*)$.
2. Minimize $\|Z - Z^*\|_F^2$; this is equivalent to maximizing $\mathrm{tr}(Y_j^T Y^* \Phi_j \Phi_j^T)$.
3. Minimize $\|Z - XW^* - \mathbf{1}b^{*T}\|_F^2$; this is equivalent to maximizing $\mathrm{tr}(Y_j^T(XW^* + \mathbf{1}b^{*T})\Phi_j^T)$.

The first and second procedures can be interpreted as rounding to the nearest point in, respectively, the Y-space (i.e. $\mathcal{Y}$) and the Z-space. The idea to round in the Z-space was originally proposed in [14]. The third procedure is a minimization of a linear upper bound of the discriminative cost $F(Y)$, where the minimization step relative to W and b has been removed. It can also be interpreted as a strongly-supervised variant of our weakly-supervised approach: once classifiers are learnt, alignment is performed. [9] reports it as giving the best results.

We will discuss this choice in more detail in section 3.3.

3 Discussions and contributions

3.1 Handling of polyphony

So far, we have assumed that only one category of event occurred at every time step, or in other words that events belonged to only one category. This was how the authors of [9] conducted their experiments.

In practice, musical events typically consist in *chords*, that is to say, in several notes occurring at once. We could represent every occurring chord in our data as a separate category, but this has two important disadvantages: first, it results in an impractical number of categories. Second, it ignores the roughly additive nature of chords: the signal corresponding to a chord is going to look fairly similar to those of each individual note. More precisely, it is common (though not universal) in the Music Information Retrieval literature (see for instance [15] and [16], both dealing with music transcription) to make those assumptions: that the signal of a chord is the sum of each component pitch's signal (additive assumption), and that each term in that sum has the same energy or L2 norm (equal energy assumption). This is equivalent to assuming that each pitch corresponds to a signal and that the signals for different pitches are decorrelated.

To account for all this, we chose to allow events to be part of several categories at once. In practice, this means that several entries in each row of Φ may be non-zero. This doesn't affect any of the calculations.

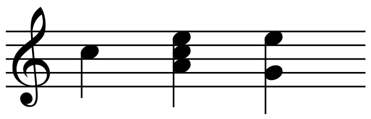
There is a choice to be made regarding the manner in which polyphony is encoded:

1. We can set every $\Phi_{i,k}$ such that note k occurs at time i to 1 (no normalization). This is consistent with the additive and equal amplitude assumptions if the data isn't normalized.

2. We can apply L1 normalization to the first representation, so that $\mathbf{1}_E\Phi = \mathbf{1}_E$. This can be thought to be more consistent with a mixture model of the sound, or with assumptions of equal amplitude, rather than equal energy.

3. We can apply L2 normalization to the first representation, so that $\operatorname{diag}(\Phi\Phi^T) = \mathbf{1}_E$. This eliminates weight differences between events in the cost function. Besides, if the data (X) is normalized, it is consistent with assumptions of additivity and of equal energy.

For a concrete exemple, consider the following snippet:



It would be encoded as such, with the columns corresponding to G, A, C, and E respectively:

First option:
$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Second option:
$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{bmatrix}$$

Third option:
$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$$

In practice, we went with the simpler first option. Both the first and third options seem to work well experimentally with normalized data, as seen for instance on figure 3.

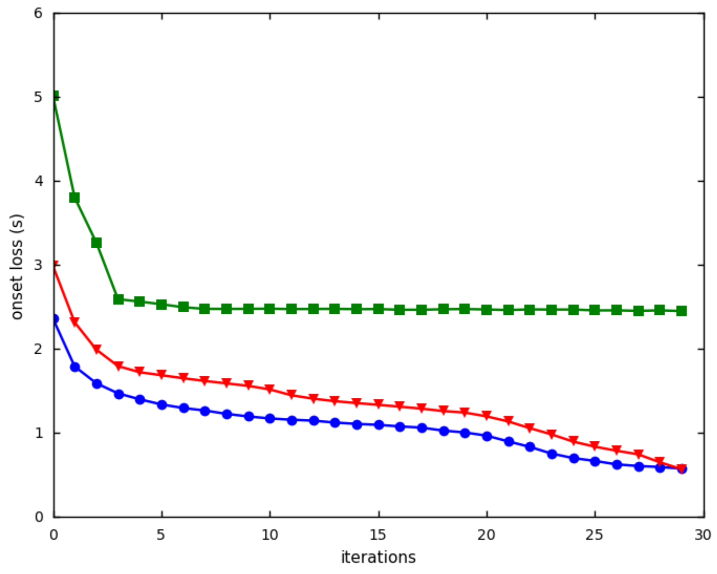


Figure 3: Evolution of the onset loss in seconds (see section 4.2.1) through iterations of the algorithm with various encodings of polyphony, using 30 recordings of Chopin's Mazurka op. 17 n$^{\text{o}}$ 4.

3.2 A generative approach: HSMMs

3.2.1 Description of the model

An alternative to the discriminative cost function we proposed is the use of a *Hidden Semi-Markov Model* (HSMM). HSMMs are an extension of HMMs relying on semi-Markov rather than Markov processes. An overview of HSMMs can be found in [17].

A discrete-time HSMM involves an underlying Markov chain going through a sequence of states $(s_k, d_k) \in S \times \mathbb{N}^*$ for $k = 0, \ldots, N-1$. s_k is the *hidden state*, and d_k is the *duration* of the k-th state. For the sake of simplicity, we will assume that the sequence of states is a time-homogeneous Markov chain by itself with transition probabilities given by $P \in |S| \times |S|$ and that the durations only depend on the associated state s, with the distribution $D(\cdot \mid s)$. Thus the Markov process follows the graphical model in figure 4.

$$\begin{tikzcd}
\text{States} & s_0 \arrow[r, "P(\cdot \mid s_0)"] \arrow[d, "D(\cdot \mid s_0)"] & s_1 \arrow[r, "P(\cdot \mid s_1)"] \arrow[d, "D(\cdot \mid s_1)"] & s_2 \arrow[r] \arrow[d, "D(\cdot \mid s_2)"] & \cdots \arrow[r] & s_{N-1} \arrow[d, "D(\cdot \mid s_{N-1})"] \\
\text{Durations} & d_0 & d_1 & d_2 & & d_{N-1}
\end{tikzcd}$$

Figure 4: The graphical model of the Markov process behind an HSMM.

The semi-Markov process in itself is defined as $s'_t = s_k$ for $t = 0, \ldots, T-1$ where:

$$\sum_{j=0}^{k-1} d_j \leq t < \sum_{j=0}^{k} d_j \tag{28}$$

$$T = \sum_{j=0}^{N-1} d_j \tag{29}$$

Thus, s' goes through the same values as s, "staying" for d_k steps on s_k. Finally, the *observations* x_t are generated at each time step t following the emission distribution $M_{s'_t}$

To summarize all of it in simpler terms, the process goes through a sequence of states; it stays in a state j for a period given by $D(\cdot \mid j)$, and then changes state following the distribution $P(\cdot \mid j)$. At each time step, a signal x is generated following a distribution M that depends on the current state solely.

The joint likelihood of a sequence of states, durations and observations can be written as follows:

$$P(\forall k, \forall t, S_k = s_k, D_k = d_k, X_t = x_t) = \prod_{k=0}^{N-2} P_{s_k, s_{k+1}} \prod_{k=0}^{N-1} D_{s_k}(d_k) \prod_{t=0}^{T-1} M_{s'_t}(x_t). \tag{30}$$

In our case, we will take the states to be the events: $S = \{0, \dots, E-1\}$. Since we know the sequence of events in advance, we do not have to consider a transition distribution P; we have $s_k = k$ for all k.[3] M is the distribution of our data features X_t given each event's pitch. The distribution D is that of the duration of each event: $d_i = t_{i+1} - t_i$; we can write the vector of those durations as $Y^T \mathbf{1}_T$. Similarly, there is a simple relation between our alignment matrix Y and the sequence s': $s'_t = \sum_i Y_{t,i}$, which can also be written as $s' = Ys$.

We choose to modelize the observation distribution M by a multivariate Gaussian distribution with covariance I_d; the expected value of each category k is denoted as q_k and we write them together in matrix form as $Q \in \mathbb{R}^{K \times d}$. We denote the category of each event i as k_i. For D, we have d_i follow a distribution proportional to $\frac{(d_i - l_i)^2}{l_i^2}$ (a discretized truncated Gaussian distribution), where l_i is the expected duration given by $\overline{Y}^T \mathbf{1}_T$. Then the joint negative log-likelihood of the signal and some alignment can be written in matrix form as:[4]

$$\begin{aligned} H(Y, Q, X) &= \frac{1}{n} \sum_{j=1}^{n} \left[\sum_{t=0}^{T_j - 1} \|q_{k_{s'_t}} - (X_j)_t\|_2^2 + \nu \sum_{i=0}^{E_j - 1} \frac{(d_i - l_i)^2}{l_i^2} \right] \\ &= \frac{1}{n} \sum_{j=1}^{n} \left[\|Y_j \Phi_j Q - X_j\|_F^2 + \nu \|(\overline{Y}_j^T \overline{Y}_j)^{-1} (Y_j - \overline{Y}_j)^T \mathbf{1}_{T_j}\|_2^2 \right]. \end{aligned} \tag{31}$$

Note that the second term is what we previously defined as h_{local}. This is why we said that our choice for h_{local} was inspired by generative models. Another choice of distribution for d_i could be used as h_{local} just as well.

Another interesting remark is that the duration penalization given by h_{local} only has to be defined on integers. We chose to have it coincide with a Gaussian distribution on reals, and in the convex relaxation of section 2.4, we extended the quadratic function over $\overline{\mathcal{Y}}$ so that the gradient would be easy to compute. We will see further along this section why we might want to pick a different extension.

3.2.2 Viterbi training of HSMMs

If we want to use the model above, we have to deal with Q. To keep our weakly-supervised approach, we'd like to learn Q and Y together without annotations:

$$\begin{aligned} &\text{minimize } \frac{1}{n} \sum_{j=1}^{n} \left(\|Y_j \Phi_j Q - X_j\|_F^2 + \nu \|(\overline{Y}_j^T \overline{Y}_j)^{-1} (Y_j - \overline{Y}_j)^T \mathbf{1}_{T_j}\|_2^2 \right) \\ &\text{s.t. } Y_j \in \mathcal{Y}_j,\ Q \in \mathbb{R}^{K \times d} \end{aligned} \tag{32}$$

[3]We essentially adopt a linear topology for our Markov chain.

[4]Up to multiplicative and additive constants.

There are two traditional ways of training HSMMs: [18] the most common one is the Expectation-Maximization algorithm (EM), which maximizes the observation likelihood $P(X)$. The other possibility, called *Viterbi training*, optimises the joint likelihood $P(X,Y)$, which is what we want to do here. It consists in alternating optimization in Q and Y:

$$Q^{(k)} = ({Z^{(k)}}^T Z^{(k)})^\dagger {Z^{(k)}}^T X \tag{33}$$

$$Y_j^{(k+1)} \in \operatorname*{arg\,min}_{Y_j \in \mathcal{Y}_j} H(Y, Q^{(k)}) \tag{34}$$

The second step uses the so-called *HSMM Viterbi algorithm* to recover the optimal alignment.[5] This algorithm is a dynamic programming method akin to DTW, and an extension of the Viterbi algorithm for HMMs. While on HMMs it runs in time $O(TE)$, it runs in time $O(ET^2)$ in the more general case of HSMMs.[6] Yet, in the case of HSMMs with log-concave duration distributions (including our pseudo-Gaussian distributions), it is possible to implement a variant of the Viterbi algorithm that runs in time $O(ET)$. To our knowledge, this was first proposed in [20], as an adaptation of earlier DNA matching techniques, and has been often ignored by later work, perhaps because it doesn't come from the inference community. We describe it in the next section.

3.2.3 A linear-time Viterbi algorithm for HSMMs

The Viterbi algorithm is used to find maximum-likelihood paths in HSMMs. We consider here the problem of minimizing $\mathrm{tr}(Y^T C) + L(Y^T \mathbf{1}_T)$, where $L(\Delta) = \sum_{i=0}^{E-1} L_i(\Delta_i)$ is a sum of convex functions of each component. The following discussion will use the same notations as those of DTW (section 2.4.1): C is the cost matrix, with dimensions $T \times E$, and $D_{t,i}$ represents the minimal cost to reach event i at time t.

The observation of [20] is the following: given that the duration distribution is concave, it is possible to predict at which point a certain subpath will become optimal, eliminating the need for later checks. In the "regular" Viterbi algorithm, at every step, the following minimization is performed:

$$D_{t,i} = \min_{t'<t} \underbrace{D_{t'-1,i-1} + \sum_{s=t'}^{t-1} C_{s,i} + L_i(t-t')}_{J_{t',i}(s)} . \tag{35}$$

Because L_i is a convex function, the quantity $L_i(t-t') - L_i(t-t'')$ (and therefore $J_{t',i}(t) - J_{t'',i}(t)$) is decreasing with t if $t' < t''$, and there is a tipping point t^* at which the path that goes to i at time t'' becomes more performant than that from time t' (see figure 5). If we calculate this tipping point, we know in advance when will have to consider each path in future steps, and no linear-time minimization is necessary. In the general case, this computation consists in finding the sign change of an univariate monotonous function on a finite set, which can be done in logarithmic time by bisection. In our heteroscedatic Gaussian case, we can actually compute

[5] More commonly referred to as *segmentation* or *path* in this context.

[6] In general, the basic HSMM Viterbi algorithm runs in time $O(NTDK)$, where N is the number of states, K their maximum incoming degree, and D the maximum duration of a state. The authors of [19] reduce this to $O(NT(D+K))$. Since the sequence of states is exactly known in our case, $K=1$ and $N=E$. We do not make any assumption on D beyond the obvious $D \leq T$.

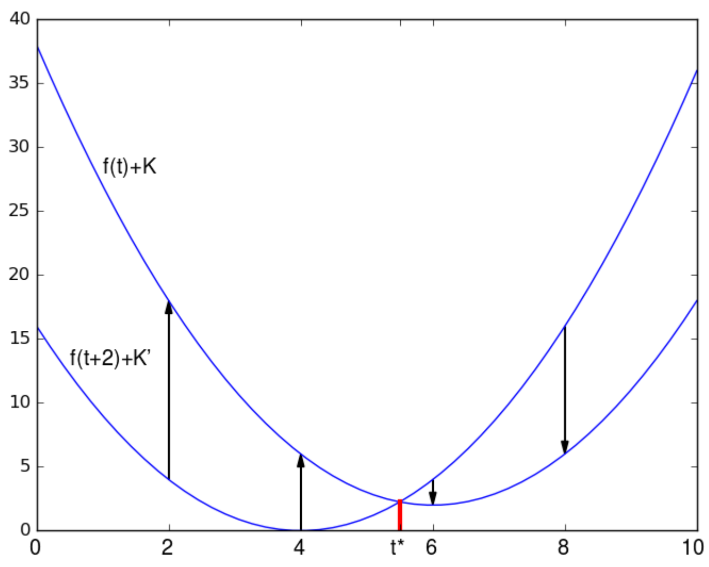


Figure 5: The difference between a convex function and the same function shifted backwards is decreasing.

t^* through a closed form in constant time. Recall that $L_i(t-t') = \frac{(t-t'-l_i)^2}{l_i^2}$. We have for $t' < t''$:

$$\begin{aligned} J_{t',i}(t) < J_{t'',i}(t) &\iff L_i(t-t') - L_i(t-t'') < \overbrace{D_{t''-1,i-1} - D_{t'-1,i-1} + \sum_{s=t'}^{t''-1} C_{s,i}}^{A} \\ &\iff (t''-t')(2t-t'-t''-2l_i) < l_i^2 A \\ &\iff t < \frac{1}{2}(t'+t'') + l_i + \frac{l_i^2 A}{2(t''-t')} = t^*. \end{aligned} \tag{36}$$

Thus we can define a function $\textsc{TippingPoint}_i$ that computes t^* from A, t' and t''.

In practice, a queue is used to store the future optimal paths for each event. The procedure is outlined in more detail in algorithm 1. Because only one element can be added to each queue at every step in the main loop, the total running time is $O(TE)$.

3.3 Revisiting rounding

The rounding procedures described in 2.4.3 and used in [9] can be quite dissatisfying in practice, as they do not incorporate the regularizing information brought by the priors. Unfortunately, it is impossible to minimize the local penalization on $\mathcal{Y}$ using DTW, as, to the contrary of the global one, it doesn't reduce to a linear form.

However, the local penalization can be interpreted, as we said, as the duration log-likelihood associated with a Gaussian HSMM. Thus, the Viterbi algorithm for HSMMs (described just above) can be used to recover the optimal path. This yields a more efficient rounding procedure, where we minimize the following objective:

$$\|Y\Phi - XW^* - \mathbf{1}_T b^{*T}\|_F^2 + h(Y). \tag{37}$$

This can be done in time $O(TE)$ using algorithm 1.

Like the third procedure of section 2.4.3, this procedure can be interpreted as an upper bound over the whole cost function, and not just the discriminative cost, that doesn't optimize over W and b. It can also be interpreted as a strongly-supervised segmentation that incorporates regularizers. More precisely, it can be seen as the optimization of a certain HSMM.

Algorithm 1 The Linear-time Viterbi algorithm: minimizing $\mathrm{tr}(Y^T C) + L(Y^T \mathbf{1}_T)$.

```
$D \leftarrow +\infty_{T \times E}$    ▷ $D_{t,i}$ is the minimal cost to reach event $i$ at time $t$.
$D_{0,0} \leftarrow C_{0,0} + L_0(1)$
Create an empty queue $Q^{(i)}$ for each event $i$.    ▷ $Q^{(i)}$ contains pairs $(t, t^*)$ of an origin time and a tipping point.
for $t = 1, \ldots, T_j - 1$ do
    $D_{t,0} \leftarrow \sum_{s=0}^{t} C_{s,0} + L_0(t+1)$
    for $i = 1, \ldots, E_j - 1$ do
        Remove entries at the front of $Q^{(i)}$ if we reached the next entries' tipping point.
        while $Q^{(i)}$ isn't empty do
            Set $(t_0, t_0^*)$ to the last element of $Q^{(i)}$.
            $t^* \leftarrow \textsc{TippingPoint}_i(D_{t-1,i-1} - D_{t_0-1,i-1} + \sum_{s=t_0}^{t-1} C_{s,i}, t_0, t)$
                ▷ $\textsc{TippingPoint}_i(A, t', t'')$ finds the zero of $L_i(t - t'') - L_i(t - t') + A$.
            if $t^* \geq T$ then    ▷ This path is never optimal.
                Exit loop.
            else if $t^* > t_0^*$ then    ▷ This path will become optimal at some point.
                Push $(t, t^*)$ at the end of $Q^{(i)}$ and exit loop.
            else if $t^* \leq t_0^*$ then    ▷ This path will become optimal but the one from $t_0$ won't.
                Remove $(t_0, t_0^*)$ from $Q^{(i)}$.
        if $Q^{(i)}$ is empty then    ▷ This path is optimal right now.
            Push $(t, t)$ into $Q^{(i)}$.
        Set $(t_0, t_0^*)$ to the first element of $Q^{(i)}$.
        $D_{t,i} \leftarrow L_i(t - t_0 + 1) + D_{t_0-1,i-1} + \sum_{s=t_0}^{t} C_{s,i}$
Backtrack through $D$ to recover $Y$.
```

This analogy leads us to a further variant consisting in repeating the optimization several times, alternating computations of W^* and b^* and of Y to refine the result. We refer to this as "alternating optimization". This is very similar to so-called "Viterbi training" of an HSMM [18], which we will describe in section 3.2.2.

We compared experimentally the three procedures in section 2.4.3, the one above and the alternating optimization, and found that both procedures taking regularizers into account yielded better results. Example results from such an experiment are shown in figure 6.

3.4 Generalized Conditional Gradient

The *Generalized Conditional Gradient* algorithm (GCG) is an extension of the Frank-Wolfe method in the case where we know how to minimize exactly part of the objective. Its name stems from the fact that Frank-Wolfe is also known as the Conditional Gradient algorithm. [13] In the Frank-Wolfe method, new search directions are found by solving the following problem:

$$x'^{(k)} \in \arg\min_x \; x^T \nabla f(x^{(k)}). \tag{38}$$

In the setting of GCG, we consider that the objective function has the form $f(x) = g(x) + h(x)$, where we know how to minimize h exactly. Thus, the problem we solve to find a search direction becomes:

$$x'^{(k)} \in \arg\min_x \; x^T \nabla g(x^{(k)}) + h(x). \tag{39}$$

Using $\gamma_k = \frac{2}{k+1}$ as step size, the GCG algorithm has the same convergence properties as Frank-Wolfe, [13] though it often performs better in practice.

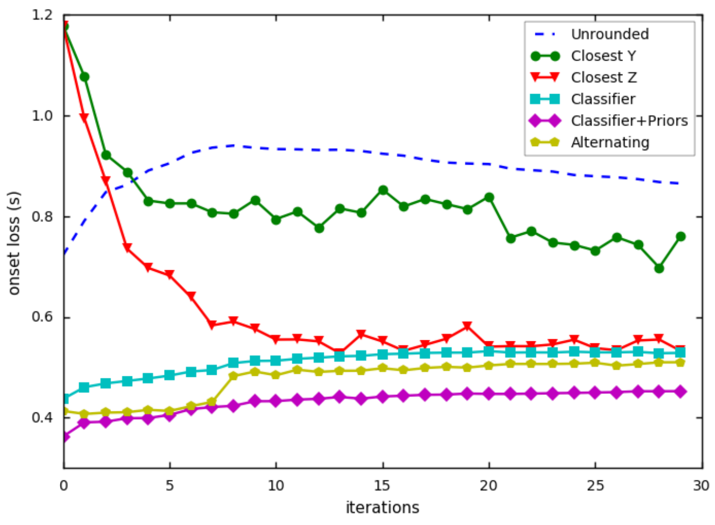


Figure 6: Evolution of the onset loss in seconds through iterations for various rounding schemes, using 100 distorted MIDI files from the Finnish Folk Tunes dataset as data. The "alternating" method was applied with 3 successive optimization steps. The "unrounded" data represents the value of h_{global} on the original point in $\overline{\mathcal{Y}}$.

The GCG method doesn't apply readily to our problem as described above, since we do not know how to minimize h_{local} on the convex domain $\overline{\mathcal{Y}}$, but only on $\mathcal{Y}$. Yet, as we mentioned in section 3.2.1, our choice of extending h_{local} over $\overline{\mathcal{Y}}$ as a smooth quadratic function is arbitrary. Instead, we can choose our extension $\hat{h}_{\text{local}}$ to be piecewise linear, so that:

$$\min_{Y\in\overline{\mathcal{Y}}} \hat{h}_{\text{local}}(Y) = \min_{Y\in\mathcal{Y}} \hat{h}_{\text{local}}(Y). \tag{40}$$

$\hat{h}_{\text{local}}$ can be defined explicitly as the convex hull of h_{local}, i.e. the greatest convex function (on $\overline{\mathcal{Y}}$) minimizing h_{local} on $\mathcal{Y}$. This function is piecewise linear because $\mathcal{Y}$ is discrete (for an intuition of what such a function looks like, see figure 7). $\hat{h}_{\text{local}}$ is an upper bound of h_{local} and coincides with it on $\mathcal{Y}$. Both can be considered to be natural yet arbitrary convex extensions of the HSMM likelihood. An advantage of this second approach is that it doesn't rely on the choice of Gaussian distributions as a local prior. Any other log-concave distribution could be extended over $\overline{\mathcal{Y}}$ in the same manner, and then optimized with the method above.

By contrast, a disadvantage of this is that such an extension is very hard to compute explicitly on all of $\overline{\mathcal{Y}}$. In fact, on a set as large as $\mathcal{Y}$, we have to renounce having an explicit form for the objective entirely. This means, amongst other things, that we can no longer perform an approximate line search, let alone one in closed form, and we have to use an "universal" step size $\gamma_k = \frac{2}{k+1}$.

As far as the global penalization is concerned, its convex hull can be known explicitly, since it is simply the extension of its linear form expression on $\mathcal{Y}$:

$$\forall Y_j \in \overline{\mathcal{Y}_j},\ \hat{h}_{\text{global}}(Y_j) = \operatorname{tr}\left(Y_j^T\left(\operatorname{diag}(L_j^T L_j)\mathbf{1}_{E_j}^T - 2\overline{Y}_j L_j^T L_j\right)\right). \tag{41}$$

We can define for convenience the following functions:

$$\hat{h} = \mu\,\hat{h}_{\text{global}} + \nu\,\hat{h}_{\text{local}} \tag{42}$$

$$\hat{G} = F + \hat{h} \tag{43}$$

We can then use the GCG algorithm to minimize $\hat{G}$. This new version of our method is very similar to the original Frank-Wolfe algorithm, except that the dynamic optimization step (26)

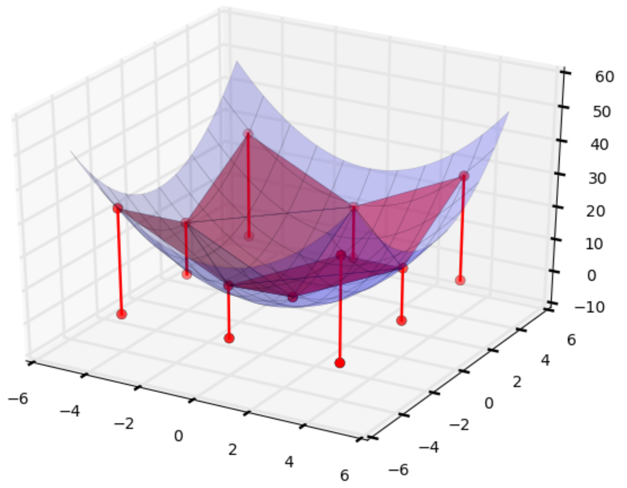


Figure 7: Illustration of the convex hull approximation (in red) of the 2D quadratic function $\|x\|_2^2$ (in blue). A simple 3×3 grid plays the role of the polytope $\mathcal{Y}$.

is replaced by:

$$Y_j'^{(k)} \in \underset{Y_j \in \mathcal{Y}_j}{\arg\min} \left[\operatorname{tr}\left(Y_j^T \left(\nabla_{Y_j} F(Y^{(k)}) \right) \right) + \hat{h}(Y) \right]. \tag{44}$$

which we can solve in time $O(T_j E_j)$ with the linear-time Viterbi algorithm.

4 Experimental discussion

4.1 Description of baseline methods

We compared our method to several other approaches to the problem of music alignment.

4.1.1 Change-point detection

Change-point detection (CPD) seeks to segment a time series in contiguous clusters. The desired number of clusters can be known in advanced or detected by the algorithm.

Using optimal Gaussian likelihood (or equivalently, variance) as a cost function, the objective minimized by CPD algorithms for a fixed number E of clusters is:

$$F_{\text{CPD}}(Y) = \|X - Y(Y^T Y)^{-1} Y^T X\|_F^2. \tag{45}$$

Note that this can be interpreted as applying the Viterbi algorithm to a linear HSMM like the one in section 3.2.1, but with uniform duration distributions and no redundancy informations (each event has its own emission distribution).

A straightforward dynamic programming algorithm can solve the problem in quadratic time, which is what we used. CPD is discussed in a more elaborate fashion in [21] and [22].

Note that CPD doesn't make any use of the information we have access to through the music score, i.e. the redundancy information from the template and the expected durations.

4.1.2 Alternating minimization

Recall that in section 3.3, we suggested alternating minimization as a rounding procedure to recover alignments in $\mathcal{Y}$ from point in $\overline{\mathcal{Y}}$.

An alternative approach consists in abandoning the convex relaxation altogether, and applying alternating optimization to the original minimization problem:

$$\begin{aligned}\text{minimize } & \frac{1}{n}\sum_{j=1}^{n}\left(\|Y_j\Phi_j - X_jW - \mathbf{1}_T b^T\|_F^2 + \mu\|(Y_j - \overline{Y}_j)L_j^T\|_F^2 + \nu\|(Y_j - \overline{Y}_j)^T\mathbf{1}_T\|_2^2\right) + \lambda\|W\|_F^2 \\ & \text{s.t. } Y_j \in \mathcal{Y}_j,\ W \in \mathbb{R}^{d\times K},\ b \in \mathbb{R}^K\end{aligned} \tag{46}$$

Instead of eliminating W and b to find a quadratic function of Y, we can perform alternating linear minimizations in (W, b) and Y. Specifically, we go through the following iterations:

$$W^{(k)} = \Gamma Z^{(k)} \tag{47}$$

$$b^{(k)} = \frac{1}{T}(Z^{(k)} - XW^{(k)})^T\mathbf{1}_T \tag{48}$$

$$C_j^{(k)} = \mathbf{1}_{T_j}\operatorname{diag}(\Phi_j\Phi_j^T)^T - 2X_jW^{(k)}\Phi_j^T - 2\mathbf{1}_{T_j}b^{(k)^T}\Phi_j^T + \operatorname{diag}(L_j^TL_j)\mathbf{1}_{E_j}^T - 2\overline{Y}_jL_j^TL_j \tag{49}$$

$$Y_j^{(k+1)} \in \operatorname*{arg\,min}_{Y_j\in\mathcal{Y}_j} \operatorname{tr}(Y_j^TC_j^{(k)}) + \nu\|(\overline{Y}_j^T\overline{Y}_j)^{-1}(Y_j - \overline{Y}_j)^T\mathbf{1}_T\|_2^2 \tag{50}$$

where Γ is defined by (7). The minimization in the fourth line can be performed with the Viterbi algorithm.

This has no convergence guarantees.

4.1.3 Viterbi training

As we have discussed, a more traditional approach to the problem would be to use a generative model. Viterbi training of an HSMM is described in section 3.2.

We can deviate from section 3.2 and include the global prior in the cost function as a linear term, though its probabilistic interpretation isn't obvious. Since the cost function of this model is very different from that used in our discriminative methods, we have to set the hyperparameters μ and ν differently. In practice we found that setting $\mu = 0$ doesn't worsen results significantly.

4.1.4 Testing time: applying an existing classifier

The methods described above (apart from CPD) attempt to simultaneously learn the model and perform the alignment. If we only want to do the latter, we can easily use the Viterbi algorithm to minimize the cost function with precomputed values of W and b. This procedure can be used as some sort of "testing phase" for our method.

If we use the ground truth data to learn W and b, we are essentially turning back to a strongly-supervised approach. Alternatively, we can learn classifiers from the expected alignment (i.e. make the assumption that the score is a good enough approximation of the ground truth); this would be equivalent to stopping alternating optimization after the first iteration.

4.2 Experimental settings

4.2.1 Performance metric

To evaluate the quality of our alignments, we use the "onset loss" metric. [9] If $(t_0, \ldots, t_{E-1})$ are the onset times of events following alignment Y, and $(t'_0, \ldots, t'_{E-1})$ those from alignment Y', this metric is defined as:

$$\delta(Y, Y') = \frac{1}{E}\sum_{i=0}^{E-1}|t_i - t'_i|. \tag{51}$$

As already noted in (16), this is similar to the global penalization:

$$h_{\text{global}}(Y) = \frac{1}{n}\sum_{j=1}^{n} \|(Y_j - \overline{Y}_j)L_j^T\|_F^2 = \frac{1}{n}\sum_{j=1}^{n} E_j \delta(Y_j, \overline{Y}_j). \tag{52}$$

We will express onset loss in seconds rather than time steps to make our results more relatable.

4.2.2 Data and features

Our experiments use two datasets: a collection of short, monophonic Finnish folk songs in MIDI format, and the sheet music and a large number of recordings of Frédéric Chopin's Mazurkas for piano, which are polyphonic pieces. Those are described in more detail in section 4.4.2.

To get interesting sound signals from MIDI data, we keep the approach of [9] of randomly distorting the local tempo, and then generating sound output. The waveforms (both those generated from MIDI and original recordings) are then transformed into vector data by computing mel-spectrograms with 40 coefficients in a rolling window with time step 0.08s.[7] L2 normalization of data points was found to improve results significantly. This isn't surprising, since our modelization of redundancy using Φ relies on the assumption that similar events occurring at different times or in different pieces will exhibit similar features.

For all experiments, we use the "expected alignment", i.e. the music score as our initialization point.

4.3 Parameter tuning

There is no obvious way to set μ and ν. We had to perform a grid search to that purpose, a rather slow and cumbersome process, and a clear disadvantage of the model. Fortunately, similar results were found regardless of the data used. An example is shown in figure 8.

In general, a value of ν too small will result in degenerate solutions of the kind typical of discriminative models (with a small number of overly big clusters), while one too big will entail that the expected alignment minimize the cost, making the model hardly informative. μ appears to be useless when applied to genuine recordings, and it can be set to zero without changing the results much. It is more important when we use distorted MIDI data in the manner of [9], where tempo variations can be much more discontinuous (since the local prior penalizes locally irregular tempi).

4.4 Experiments and comparison

4.4.1 Preliminaries

We applied our method the different techniques described in section 4.1 to different pieces, and compared the results in terms of onset loss and objective convergence.

In the graphs of figures 9 and 10, the following identifiers describe the methods:

- GCG: the GCG algorithm (section 3.4). GCG-RD denotes the point obtained after the rounding procedure of section 3.3.
- FW: the Frank-Wolfe algorithm (section 2.4). FW and FW-RD follow the same convention as above.

[7] We experimented with other values for the various numeric parametres, as well as other representations altogether such as MFCCs, and found no significant difference in final results. We are of course limited by processing time and memory requirements.

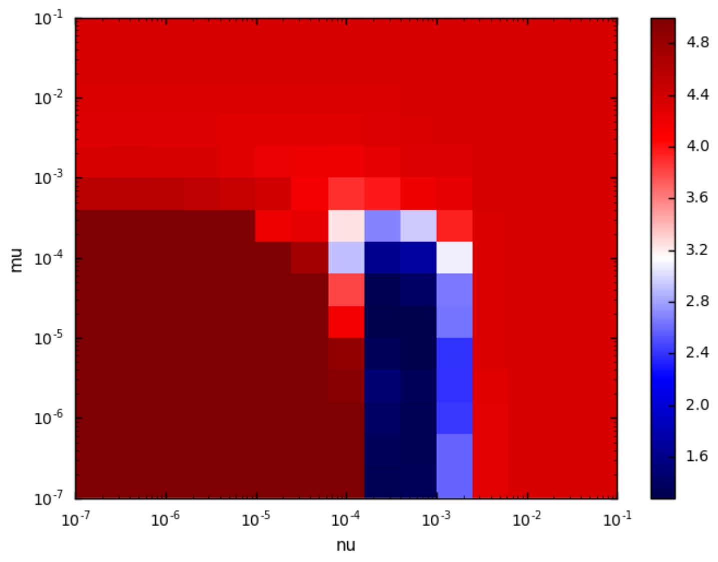


Figure 8: Results (onset loss in seconds) of a grid-search tuning of the parameters on 20 recordings of Chopin's Mazurka 17-4. μ is the y-axis, and ν the y-axis. The data was capped at 5 for greater visibility; the onset loss gets much greater when μ and ν are close to 0.

- Alt: alternating optimization (section 4.1.2).
- Viterbi: Viterbi training (section 3.2.2).
- Score: the unoptimized expected alignment.
- CPD: Change-point detection (section 4.1.1).
- ClScore: using a classifier learnt from the score (section 4.1.4).
- ClGT: the same thing using ground truth data, i.e. the strongly-supervised approach. Intuitively it should be impossible to do better than this.

The RD suffix indicates rounding of elements of $\overline{\mathcal{Y}}$ to $\mathcal{Y}$ at each step.

4.4.2 Description of datasets

Corpora of well-annotated recorded music are relatively hard to come by, precisely because we lack good audio-to-score alignment methods to generate them automatically. We mainly used two datasets for our experiments.

The *Finnish folk songs database*[8] contains about 7000 simple monophonic tunes with length a few seconds to one minute, in a simple matrix format that is more or less equivalent to basic MIDI. Those tunes were originally compiled by scholars in the early 20th century, before being translated to a digital format by the authors of [23] in 2004.

The *Mazurka project*[9] consists in a compilation of about 3000 recordings of Chopin's 59 Mazurkas, established by the UK's CHARM project. For about 400 of those recordings, "reverse conducting" annotations were manually added, which means that somebody manually marked the onset times of every note on the score in the recording, providing us with ground truth data for our experiments. Our experiments focussed on three Mazurkas that came with several dozens annotated recordings, the mazurkas op. 17 n° 4, op. 24 n° 2 and op. 68 n° 3. Collectively

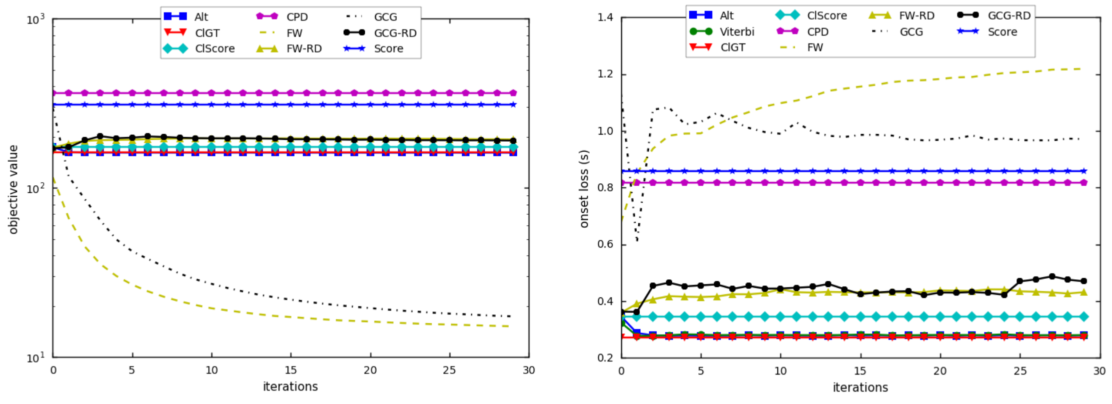
[8] Available at http://esavelmat.jyu.fi//collection_download.html.

[9] http://www.mazurka.org.uk/.

the Mazurkas have almost all 81 pitches of the piano, but a single one typically has around 30-40 different pitches only. Most recordings have lengths in the 2-5 minutes range.

4.4.3 Monophonic experiments

Our first experiments used the Finnish folk songs dataset, and mostly follow the experiments in [9]. We randomly picked 100 of them, amounting to a bit less than 1 hour of MIDI data, and applied Gaussian distortion to note durations, in the same manner as [9]. We then generated a waveform from the symbolic data, and added Gaussian noise to it to make the data more similar to real musical recordings. The original rhythm was used as an "expected" alignment $\overline{Y}$. The results are shown in figure 9.



(a) (b)

Figure 9: Optimization of various methods on 100 distorted Finnish folk songs. (a) represents the value of the objective function G, and (b) that of the onset loss compared to ground truth. Recall that GCG doesn't actually optimize G, but $\hat{G}$ instead, and that we cannot compute $\hat{G}$ explicitly.

Apart from CPD, all models were found to recover some of the information. This means that the knowledge of the music score is genuinely helpful. Discriminative models perform about as well as Viterbi training, but our convex relaxation doesn't seem very useful as the simpler and faster method of alternating optimization yields both better minimizers of the objective in $\mathcal{Y}$ and a smaller onset loss.

Both choices of convex relaxation have similar performances, and G appears to be reasonably close to $\hat{G}$.

4.4.4 Polyphonic experiments

The Finnish folk tunes are not very realistic data, as the MIDI output is much simpler in form than actual recordings, and our tempo distortion doesn't resemble that of a real musician interpreting a piece. We conducted a more informative experiment using recordings of Chopin's Mazurkas. The results are shown in figure 10.

The results are similar to those on Finnish folk songs. GCG now outperforms FW in terms of onset loss; this appears to be due to the greater importance of the local penalization in this setting. Since the local penalization has low rank, first order approximation of the quadratic form does a poor job of minimizing it, whereas the piecewise linear version can be optimized exactly. In general, discriminative methods still outperform Viterbi training; this is much clearer on a more difficult problem like Mazurka 17-4, which has lots of *rubato* (local tempo variation), than on an easy one like the very regular Mazurka 24-2. Alternating minimization, which we

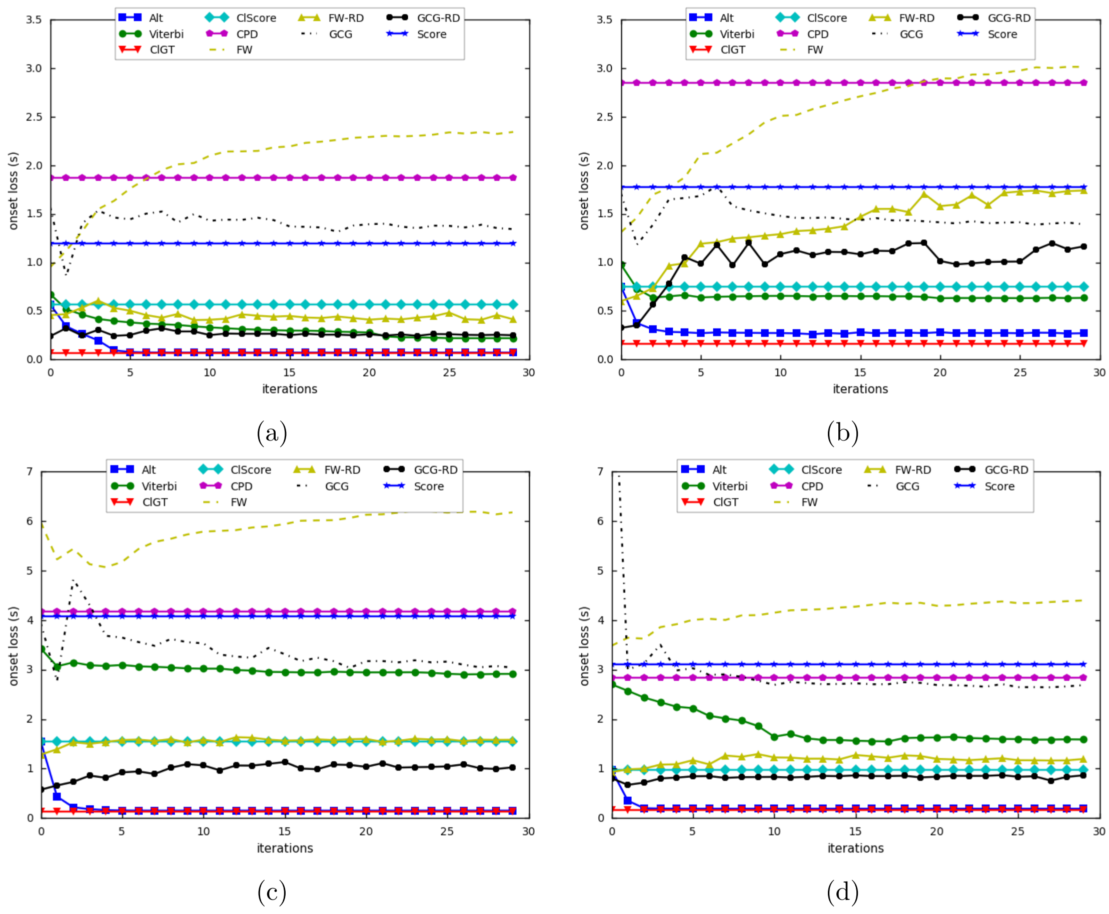


Figure 10: Optimization with various methods on 30 recordings of the mazurkas (a) 24-2 ($K = 50$), (b) 68-3 ($K = 34$), (c) 17-4 ($K = 40$) and (d) a random selection of all Mazurkas ($K = 62$).

originally conceived as a rounding procedure, is much more effective than convex relaxation in all cases.

As planned, the concavity of the Gaussian distribution leads to regular spacing between identical consecutive events. This may or may not reflect the actual interpretation but it is rarely very far from it. Since none of our methods incorporate a real modelization of tempo, they react poorly to durable tempo changes such as the one in the middle of mazurka 68-3 (made even harder by a long series of similar events, see figure 11). This explains the poor performances of all methods on that piece.

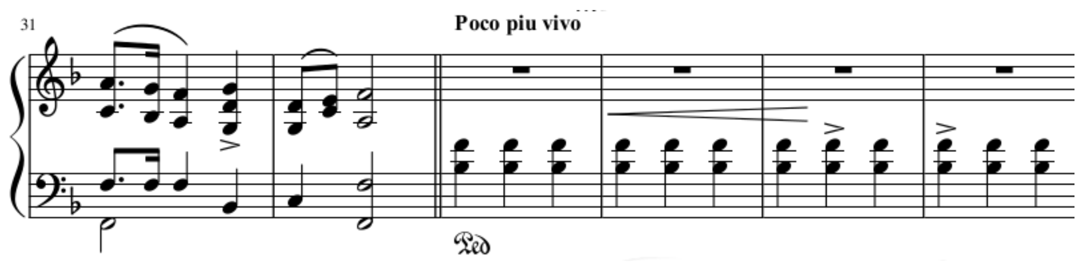


Figure 11: This tempo change in Mazurka 68-3 confuses our method.

4.4.5 Summary

Table 1 has the results for all the experiments described above.

	Score	CPD	ClScore	ClGT	Viterbi	Alt	FW	GCG
Finnish folk	0.97	1.04	0.37	0.34	0.57	0.40	0.46	0.48
Mazurka 17-4	4.07	4.17	1.55	0.14	2.92	0.15	1.58	1.03
Mazurka 68-3	1.77	2.85	0.75	0.16	0.64	0.27	1.74	1.17
Mazurka 24-2	1.20	1.87	0.57	0.07	0.22	0.07	0.41	0.25
All mazurkas	3.11	2.83	0.98	0.16	1.59	0.19	1.20	0.87

Table 1: Onset loss in seconds for various settings and methods.

5 Conclusion and perspectives

In this work, we have discussed and evaluated various improvements and elaborations on the alignment method proposed by [9]. This weakly-supervised approach has the advantages that it doesn't need fully annotated data for training, and at the same time, it doesn't rely on pre-trained models for classification.

Our experiments have shown the original convex relaxation as well as our variant suitable for the GCG algorithm perform acceptably on real-world polyphonic data, and widely outperform HSMMs used in a similar fashion. From an optimization perspective, this further encourages the use of discriminative costs for segmentation and clustering problems, and suggests that convex relaxations of discriminative problems based on the assignment matrix Z rather than the equivalence matrix M (cf section 2.2.3) could yield fruitful results for other problems. It should be noted, though, that the quality of our segmentations is largely dependent on our regularization – specifically, on what we call the "local" penalization – which in turn is based on traits unique to musical data. Application to other problems is dependent on a appropriate choice of regularization.

From the perspective of audio-to-score alignment, we found that our convex relaxations were widely outperformed by a simpler alternating optimization, perhaps disappointingly. The two methods may be combined if we use alternating optimization as our rounding procedure, though in this specific instance we found no practical advantage in doing so. Results do validate our choice of cost function and shows the potential of well-regularised discriminative models. Our approach makes a number of choices for which alternatives should probably be investigated: one possibility is to use different duration distributions for the local prior. While it does a good job of eliminating degenerate solutions, our Gaussian prior is too strict in practice and makes it impossible to detect large deviations from the score; a distribution with heavier tails might perform better. Our alternative convex relaxation does aways with the need for a smooth prior with an easy-to-compute gradient, further extending the range of choices. Another possibility of improvement would a more sophisticated modelization of the relation between features and templates, such as the cost function of [12], though this would make optimization harder, or more ambitiously, a radically different choice of regularization that would incorporate some form of tempo modelization, taking inspiration from generative models that treat the tempo as a latent variable such as [7].

References

[1] Asif Ghias, Jonathan Logan, David Chamberlin, and Brian C Smith. Query by humming: musical information retrieval in an audio database. In *Proceedings of the third ACM international conference on Multimedia*, pages 231–236. ACM, 1995.

[2] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1643–1654, 2010.

[3] Roger B Dannenberg. An intelligent multi-track audio editor. In *Proceedings of international computer music conference (ICMC)*, volume 2, pages 89–94, 2007.

[4] Andreas Arzt, Gerhard Widmer, and Simon Dixon. Automatic page turning for musicians via real-time machine listening. In *ECAI*, pages 241–245, 2008.

[5] Christopher Raphael. A bayesian network for real-time musical accompaniment. In *NIPS*, pages 1433–1439, 2001.

[6] Ferréol Soulez, Xavier Rodet, and Diemo Schwarz. Improving polyphonic and poly-instrumental music to score alignment. In *International Conference on Music Information Retrieval (ISMIR)*, page 6, 2003.

[7] Arshia Cont. A coupled duration-focused architecture for real-time music-to-score alignment. *IEEE transactions on pattern analysis and machine intelligence*, 32(6):974–987, 2010.

[8] Cyril Joder, Slim Essid, and Gael Richard. Learning optimal features for polyphonic audio-to-score alignment. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(10):2118–2128, 2013.

[9] Rémi Lajugie, Piotr Bojanowski, Philippe Cuvillier, Sylvain Arlot, and Francis Bach. A weakly-supervised discriminative model for audio-to-score alignment. In *41st International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2016.

[10] Francis R Bach and Zaïd Harchaoui. Diffrac: a discriminative and flexible framework for clustering. In *Advances in Neural Information Processing Systems*, pages 49–56, 2008.

[11] Hao Cheng, Xinhua Zhang, and Dale Schuurmans. Convex relaxations of bregman divergence clustering. *arXiv preprint arXiv:1309.6823*, 2013.

[12] Armand Joulin, Jean Ponce, and Francis R Bach. Efficient optimization for discriminative latent class models. In *Advances in Neural Information Processing Systems*, pages 1045–1053, 2010.

[13] Francis Bach. Duality between subgradient and conditional gradient methods. *SIAM Journal on Optimization*, 25(1):115–129, 2015.

[14] Piotr Bojanowski, Rémi Lajugie, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. Weakly supervised action labeling in videos under ordering constraints. In *European Conference on Computer Vision*, pages 628–643. Springer, 2014.

[15] Taylor Berg-Kirkpatrick, Jacob Andreas, and Dan Klein. Unsupervised transcription of piano music. In *Advances in neural information processing systems*, pages 1538–1546, 2014.

[16] Paris Smaragdis and Judith C Brown. Non-negative matrix factorization for polyphonic music transcription. In *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.*, pages 177–180. IEEE, 2003.

[17] Shun-Zheng Yu. Hidden semi-markov models. *Artificial Intelligence*, 174(2):215–243, 2010.

[18] Jüri Lember, Alexey Koloydenko, et al. The adjusted viterbi training for hidden markov models. *Bernoulli*, 14(1):180–206, 2008.

[19] Ritendra Datta, Jianying Hu, and Bonnie Ray. On efficient viterbi decoding for hidden semi-markov models. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008.

[20] David Tweed, Robert Fisher, José Bins, and Thor List. Efficient hidden semi-markov model inference for structured video sequences. In *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 247–254. IEEE, 2005.

[21] Guillem Rigaill. Pruned dynamic programming for optimal multiple change-point detection. *arXiv preprint arXiv:1004.0887*, 2010.

[22] Rebecca Killick, Paul Fearnhead, and IA Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598, 2012.

[23] Tuomas Eerola and Petri Toiviainen. Digital archive of finnish folk tunes. *Computer Database, University of Jyvaskyla*, 2004.