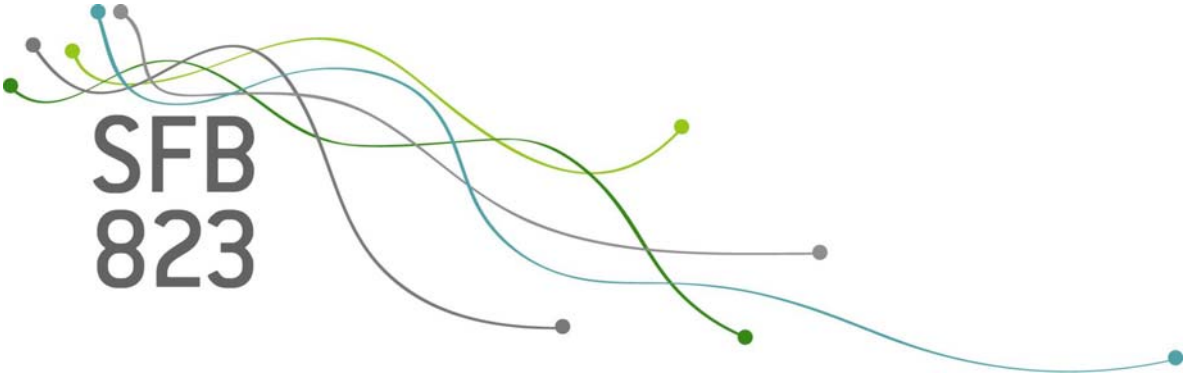
SFB 823

Discussion Paper

Model based optimization of music onset detection

Nadja Bauer, Klaus Friedrichs,
Claus Weihs

Nr. 47/2015



Model Based Optimization of Music Onset Detection

Nadja Bauer, Klaus Friedrichs, and Claus Weihs *

December 7, 2015

Abstract

In this paper a comprehensive online music onset detection algorithm is introduced where – in contrast to many other relevant publications – 14 important algorithm parameters are optimized simultaneously. For solving the optimization problem we derive an extensive tool for iterative model based optimization.

In each iteration, a very time consuming evaluation has to be performed on a large music data base. To speed up this procedure, the expected performance of each newly proposed setting is estimated in a pretest on a representative part of the data so that just very promising points are evaluated on all data. We compare different variants of the classical and the fast optimization strategies with respect to the F-values of their best identified parameter settings. The performance of the fast approach appears to be competitive with the classical one while saving more than 80% of music piece evaluations on average.

Our best found parameter settings, both for online and offline onset detection, are mainly in accordance with the usual choices in the state-of-the art literature concerning, e.g., the spectral flux detection function or preferences for window length and overlap. However, we also found unexpected results. For example, the adaptive whitening pre-processing step showed no effect.

Keywords: Onset Detection, Model Based Optimization, Instance Optimization, Kriging.

1 Introduction

A tone onset is the time point of the beginning of a musical note or another sound. Tone Onset Detection (OD) in music signals is an important step for many subsequent applications like music transcription and rhythm recognition.

*N. Bauer, K. Friedrichs and C. Weihs are with the Department of Statistics, TU Dortmund University, D-44221 Dortmund, Germany, e-mail: (bauer, friedrichs, weihs@statistik.tu-dortmund.de).

Several approaches have been proposed, but most of them can be reduced to the same basis algorithm just differing in the parameter settings ([1], [2], [3], [4]). They all follow the same scheme: windowing the signal, calculating an Onset Detection Function (ODF) for each window and localizing the tone onsets by picking the relevant peaks of the ODF. Many numerical and categorical parameters are involved in this procedure like the window size, the window overlap and the applied ODF. However, neither their influences on the algorithm performance nor their optimal values are realistically quantified. Additionally, there are many cross-dependencies, e.g., the optimal overlap between neighboring windows might be dependent on the chosen window size.

Although the necessity of parameter optimization arises in nearly all studies on OD, they usually just examine a subset of all possible parameters, while other parameters are set to some fixed empirical values, which were determined in preview studies or are frequently used in the OD community. [3] optimizes one numerical thresholding parameter fixing all further parameters. In [2] 6 ODFs are compared while other OD parameters are fixed. [4] identifies 4 influential factors (each with two levels) for the peak selection strategy (like the kind of the moving function or the normalization type) and applies 5 of the possibly $2^4 = 16$ combinations to 6 ODFs. The comparisons are done on a data set of 23 music pieces.

This leads to the further problem that these optimizations are just conducted on rather small data bases and the results are not validated on an additional test data base. For example, [2] draws the attention to over-fitting and other problems occurring in onset detection tasks.

In any case, these local optimizations are suboptimal due to cross-dependencies of parameters which may yield a true optimum in an area just not examined. Here, the next challenge arises: how to handle the huge parameter space for which a naive grid-search would be too time consuming. An approved method for such optimization tasks is Model Based Optimization (MBO) ([5], [6]). After the initial phase, i.e., an evaluation of some randomly chosen starting points, new points are proposed and evaluated iteratively with respect to a surrogate model fitted to all previous evaluations and an appropriate infill criterion to decide which point is the most promising. The most prominent infill criterion is expected improvement which looks for a compromise of surrogate model uncertainty in one point and its expected positive improvement compared to the current optimum.

One specific characteristic of OD optimization is that each parameter setting is rated by its average performance over all music pieces in an appropriate data base. For the purpose of better generalization, these music pieces will also be called problem instances in the following. Note that instance-based optimization is also applicable to many other domains. In [7] we introduced FMBO, an approach to speed-up MBO for instance-based optimization problems using the idea that instead of evaluation of all instances, parameter settings might already be recognizable as unpromising after evaluation of just a small subset of instances. Therefore, a subset of representative instances is chosen which is used in the consecutive stages for the classification of the proposed parame-

ter settings into "good" and "bad" settings. Only if the probability of beating the current optimum is above a specified threshold, the setting is classified as "good" and evaluation is continued on all instances.

For an adequate performance prediction, the representative subset of instances should be as diverse as possible. This is achieved by clustering the instances with respect to the difficulty of finding onsets, a concept which is further improved in this study. Additionally, in contrast to our previous study in [7], the number of parameters of the optimized algorithm is enlarged. Among other things, the extended algorithm also enables online variants of OD. Furthermore, the optimization is conducted on a larger data base combining almost all data sets frequently used in previous studies. Validation of the different approaches is conducted in a more sophisticated manner by dividing the data base into training and test data. Finally, dummy Kriging is tested and compared as an alternative surrogate model.

In Section 2 we briefly describe the OD algorithm with the 14 parameters to be optimized. Note that the algorithm is able to perform online OD when setting two parameters (time limits) to very small values. For this reason, we will optimize online and offline OD separately. The extensive data base consisting of many data sets used in previous studies is introduced in Section 3. The comprehensive MBO tool is presented in Section 4. The instance based method FMBO for speeding up classical MBO and its new advances are introduced in Section 5. Section 6 describes the validation procedure for comparing the optimization strategies and defines the comparison experiments. Afterwards, the results of these experiments are analyzed followed by a short discussion about the best OD parameter settings for online respectively offline OD. Finally, in Section 7 the main findings are summarized and several ideas for future research are discussed.

2 Onset Detection Algorithm

In this section, we will explain the individual steps of the following scheme of classical onset detection:

[A.] Split the signal into small (overlapping) windows.
[B.] Pre-process the data.
[C.] Compute an ODF.
[D.] Normalize the ODF.
[E.] Threshold the normalized ODF.
[F.] Localize the tone onsets.

We will focus on the content-related meaning of a step or a feature and will waive formulas if they can be easily found in the cited literature. For the $d = 14$ parameters of the following optimization (N, h, *window.fun*, α, m, r, *odf.fun*, *moving.fun*, λ(or p), $t(r_T)$, $t(l_T)$, $t(r_O)$, $t(l_O)$, $t(min.dist)$) permitted discrete values or ranges will be given. Beside the established methods in [1], [4], and [2] we will also introduce our own extensions.

2.1 Windowing and the STFT

We assume a digital audio signal sampled with a rate of $F_s = 44.1$ kHz. This signal is split into l (overlapping) windows of length N. Since we intend to carry out a Short-Time Fourier Transformation (STFT) in each of these windows, powers of 2 are chosen as window lengths. We will consider $N = 512, 1024, 2048$ and 4096. The hop size parameter h determines the distance in samples between the windows. We vary the hop sizes between $h = N/2$ and N.

In each window a window function is applied to weight the samples. See [8] for an overview of such functions. We compare the four most frequently used window functions, namely the uniform, Hamming, Blackmann, and Gauss (with standard deviation $\sigma = 0.4$) functions. The parameter *window.fun* represents the type of window function in our optimization. Finally, an STFT ([2]) is carried out in each window.

2.2 Pre-Processing the Signals

Adaptive whitening is proposed as a pre-processing method by [9]. This leads to a signal based re-weighting of the STFT so that the activity variations of the different frequencies are mapped to a similar range. The method operates in online manner and depends on two parameters. The memory parameter $m \in [0, 1]$ influences the time for forgetting past signals. The greater m, the longer is the memory. The rounding parameter r depends on the permitted signal amplitude, where values greater than the spectral energy maximum switch off adaptive whitening. Here, r lies in $[0, 10^7]$.

2.3 Onset Detection Functions

The computation of an onset detection function in a window of the pre-processed signal is often called reduction, since after this step not the signal is analyzed anymore but only the ODF values. Many ODFs are based on the comparison of neighboring windows. An increase of an ODF generally indicates an onset, a decrease an offset. Also, offset information can improve onset detection [10]. Subsequently, we will briefly discuss the 18 ODFs, represented by the parameter *odf.fun* in our optimization. Note that for the features in the paragraphs a), b), and c) a transformation by STFT is not needed.

Zero Crossing Rate The *zero-crossing rate* (ZCR) gives the number of sign changes of the signal amplitude in a window. The direction of such changes is ignored. Therefore, the absolute difference of the ZCR to the previous window (*ZCR.Abs.Diff*) is of interest: the greater the difference, the greater the likelihood of an onset.

Amplitude Maximum The difference between the *absolute maxima* of neighboring windows appears to be a good indicator for an onset of string instru-

ments ([11]). In our optimization, we will consider two features, the difference (*AM.Diff*) as well as the absolute difference (*AM.Abs.Diff*).

Amplitude Energy The sum of the squared signal values represents the *energy* in a window [12]. Again, two features are built, the difference (*AE.Diff*) and the absolute difference (*AE.Abs.Diff*) of the amplitude energy in neighboring windows.

Weighted Spectral Energy Often, a tone onset can be distinctly recognized from individual frequencies, whereas other frequencies provide a blurred image. Therefore, [13] proposes a linear weighting of the absolute values of the Fourier coefficients (*High Frequency Content feature*, HFC). Again, the difference (*HFC.Diff*) and the absolute difference (*HFC.Abs.Diff*) of the linearly weighted spectral energy in neighboring windows are considered.

As an alternative method the Gauss window function is used for weighting (*GaussFrequencyContent*, GFC). The corresponding two features are then named *GFC.Diff* and *GFC.Abs.Diff*.

Spectral Centroid The *spectral centroid* ([14]) indicates the location of the spectral distribution. The direction of a change is ignored so that only the absolute spectral centroid differences in neighboring windows are considered (*SC.Abs.Diff*).

Spectral Spread The *spectral spread* ([14]) of a window represents the timbre of the playing instrument. Small values indicate instruments with only few overtones. Again, we are only interested in absolute differences of this feature in neighboring windows (*SSp.Abs.Diff*).

Spectral Skewness The *spectral skewness* ([14]) is a measure for the skewness of the frequency distribution. Low tones with few overtones will cause a positive skew. Also here we only consider the absolute differences of feature values (*SSk.Abs.Diff*).

Spectral Flux Because of its particularly good recognition rate ([2], [4], [7]) the *spectral flux* (*SF*) is one of the most popular features for onset detection. The basic idea is to sum up the positive differences of the spectral amplitudes of neighboring windows for all frequencies. Negative differences are related to tone offsets and are hence not considered. Alternatively, instead of the sum of absolute differences the Euclidean distance of the Fourier coefficients in neighboring windows can be used (*SE*, [15]).

Phase Deviation We expect that within a tone the growth of the phase between neighboring windows stays somewhat constant [16]. Therefore, the time series of the second differences of the phase is of interest. The *phase deviation*

(PD) is then defined as the mean of the absolute values of the second differences over all frequencies. [2] proposes *normalized weighted phase deviation* ($NWPD$) where the second differences are weighted by the corresponding percentage share of the absolute amplitude value regarding the signal itself.

Complex Domain The *complex domain* (CD) estimates the frequency amplitude in the actual window from the values in the two previous windows assuming a stationary signal. If the sum of the absolute differences of the estimated and the actual values over all frequencies is big, this is interpreted as an indicator for an onset (or offset). Since it is important to distinguish onsets from offsets, [2] proposes the *rectified complex domain* (RCD), where amplitude differences are only taken into account if the absolute amplitude is increasing with respect to the previous window.

2.4 Normalization

The aim of *normalization* is to transform the *odf* feature vector into a standardized form for the subsequent thresholding. First, exponential smoothing with parameter $\alpha \in [0, 1]$ can be applied, where for $\alpha = 1$ the time series stays unchanged and for $\alpha = 0$ all values of a feature are equal. The smoothed vector will be termed $\tilde{odf}$.

Most normalization methods are aiming at the scaling of $\tilde{odf}$ to a standard interval utilizing, e.g., $\max(\tilde{odf})$ and affecting the online ability of the method. In what follows, we will, therefore, introduce threshold functions working with non normalized but only smoothed features. A similar approach for online capable normalization was proposed by [17].

2.5 Threshold Function

Since not every local maximum of the *odf* vector represents an onset, the threshold function aims at the distinction between relevant and non-relevant variations. A fixed value for the threshold is unfavorable since the method could then not react to dynamic changes of the signal. Instead, *moving threshold functions* are widespread ([4]):

$$T_i = \delta + \lambda \cdot mov.fun(|\tilde{odf}_{i-l_T}|, \ldots, |\tilde{odf}_{i+r_T}|), \tag{1}$$

$i = 1, \ldots, l$, where the parameter *mov.fun* (moving function) is either the median or the arithmetic mean. l_T and r_T are the numbers of windows to the left and to the right, respectively, of the ith window which are used in the calculation of *mov.fun*. We use $\delta = 0$ since $\tilde{odf}$ was not normalized and we optimize λ in $[1.1, 2.6]$.

Following [18] we also allow a *moving p-quantile* function as the third option of the *mov.fun* parameter. However, in this case the parameter λ is fixed to 1 and p is optimized in the interval $[0.8, 0.98]$ instead.

2.6 Localization of Tone Onsets

The finally localized tone onsets should fulfill the following two conditions: $\tilde{odf}$ values should exceed the threshold being a local maximum. Following [17] we also use a third condition: A minimum distance *min.dist* (in number of windows) between the actual window and the window of the previous tone onset $i_{prev.onset}$ should be exceeded:

$$O_i = \begin{cases} 1, & \text{if } \tilde{odf}_i > T_i \text{ and} \\ & \tilde{odf}_i = \max(\tilde{odf}_{i-l_O}, \ldots, \tilde{odf}_{i+r_O}) \text{ and} \\ & i > i_{prev.onset} + min.dist, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

$\boldsymbol{O} = (O_1, \ldots, O_l)^T$ is the tone onset vector and l_O and r_O are additional parameters, namely the number of windows to the left or right of the actual window, respectively, which are used for the calculation of the local maxima.

The left limits of windows with $O_i = 1$ are taken as the time points of the tone onsets. A found tone onset is correctly identified if it is inside a tolerance interval around the true onset. We use ± 50 ms as the tolerance ([1, 2]).

2.7 Number of Windows as a Function of Time

In contrast to most papers on the topic, we do not fix N and h a priori but optimize these parameters so that setting, e.g., $r_T = 5$ could stand for very different time periods depending on window length N and hop size h. Therefore, all parameters related to numbers of windows are re-defined according to the desired time length, e.g. $t(r_T)$, and N, h. For example,

$$r_T = getNumberOfWindows(t(r_T), N, h). \tag{3}$$

The output of the *getNumberOfWindows* function is the number of windows so that the time restriction, e.g. $t(r_T)$, is not exceeded. Therefore, we will not consider the parameters r_T, l_T, r_O, l_O, and *min.dist*, but the times $t(r_T)$, $t(l_T)$, $t(r_O)$, $t(l_O)$, and $t(min.dist)$.

Since for online applications $t(r_O)$ and $t(r_T)$ should be as small as possible (to achieve minimal latency), the corresponding permitted intervals are set to [0, 0.01] s. In the offline case and universally for $t(l_O)$ and $t(l_T)$ this interval is set to [0, 0.5] s. The interval for parameter $t(min.dist)$ is [0, 0.05] s.

2.8 Evaluation of Tone Onset Detection

The goodness of the tone onset detection is measured by the F-measure taking into account the tolerance regions ([2]):

$$F = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}, \quad F \in [0, 1], \tag{4}$$

where TP, FP, and FN stand for the number of *true positive cases*, *false positive cases*, and *false negative cases*, respectively, and $F = 1$ represents an optimal detection. Note that the *true negative* cases are not taken into account in the calculation of the F-measure. Another disadvantage is that the distance between true and estimated onsets are only taken into account via the tolerance region.

An alternative evaluation measure is therefore the mean (relative) deviation. This will be called D-measure in what follows. We will optimize the F-measure and use the D-measure only as an additional evaluation feature, e.g., for the clustering in the next section.

3 Music Data Base

The greatest challenge with respect to the creation of a music data base is the necessity of information about the onset times. There are at least two ways to generate audio data with the corresponding onset times. In the literature, music pieces are manually annotated most of the time, leading to only small numbers of annotated pieces. The second possibility is the use of the MIDI format, offering the advantage to have many such music pieces at one's disposal. There are different programs available which can generate audio from MIDI using instrument specific signal models. Naturally, a music piece generated synthetically in this way will normally not realistically mimic real music recordings.

The aim of the composition of the data base for our optimization is to cover as many music aspects as possible, e.g., way of generation (annotated real music, synthesized MIDI), type of instrument, degree of polyphony, tempo, music genre, and music style. Different data bases are combined in order to cover the desired aspects. Altogether, we combine 6 data bases comprising 460 music pieces with 58545 tone onsets.

Bello data base The data base in [1] is an often cited manually annotated data base with 23 recordings and 1058 tone onsets. The author indicates that the number of tone onsets has been reduced by 7 onsets in comparison to the original paper in an extensive revision.

Holzapfel data base This is the manually annotated data base from [3]. We also use the corrections by Böck [17]. Overall, this leads to 3278 tone onsets.

Böck data base In [17], additional 208 manually annotated recordings with 23414 tone onsets appear besides the data bases of Bello and Holzapfel. From these recordings 31 music pieces are not used to reduce optimization time. This leads to 177 music pieces with 15647 tone onsets.

Folk song data base This data base was introduced in [15]. Overall, it comprises 24 music pieces with 3120 tone onsets realized from 2 MIDI folk songs played by 6 instruments in 2 tempi. The MIDI data sets were transformed to

WAV with the *RealConverter* program developed in [11] using *RWC* real music tone libraries ([19]).

Music epoch data base One part of this data base (12 pieces) was introduced in [18]. It comprises classical European music from 6 time periods: Middle Ages, Renaissance, Baroque, Classic, Romantic, and modern times. The other part is based on 10 MIDI music pieces: 6 solo pieces and 3 duets. Moreover, each of the 22 MIDI pieces is transformed to WAV once with synthetical tones (by means of a *MIDI to WAV Converter*[1]) and once with real tones (by means of *RealConverter*). This leads to 44 music pieces with 14456 tone onsets.

MIDI data base This is a data base from [7] comprising 200 music pieces. However, 100 pieces were eliminated to reduce optimization time leading to 100 music pieces with 22006 tone onsets.

4 Model Based Optimization

The aim of model based optimization (MBO) is the minimization of a time intensive and highly complex target function $f : \mathcal{X} \subset \mathbb{R}^d \to \mathbb{R}$, $f(\boldsymbol{x}) = y$, $\boldsymbol{x} = (x_1, \ldots, x_d)^T$. Each function parameter x_i is assumed to be a value of a continuous feature in a pre-fixed optimization interval $[\ell_i, u_i]$[2]. The parameter space is the cartesian product of the individual intervals: $\mathcal{X} = [\ell_1, u_1] \times \ldots \times [\ell_d, u_d]$. A possible parameter setting $\boldsymbol{x}_i \in \mathcal{X}$ is called a *point* and y_i is the target value in this point. A set of n points is called a *design* and is denoted $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^T$. Moreover, $\boldsymbol{y} = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n))^T$ is the vector of the target values on $\mathcal{D}$.

The MBO scheme has the following form:

1: generate an initial design $\mathcal{D} \subset \mathcal{X}$
2: evaluate f on the initial design: $\boldsymbol{y} = f(\mathcal{D})$
3: **while** optimization budget is not exhausted **do**
4: fit the surrogate model on $\mathcal{D}$
5: find $\boldsymbol{x}^*$ with the best infill criterion value
6: evaluate f on $\boldsymbol{x}^*$: $y^* = f(\boldsymbol{x}^*)$
7: update $\mathcal{D} \leftarrow (\mathcal{D}, \boldsymbol{x}^*)^T$ and $\boldsymbol{y} \leftarrow (\boldsymbol{y}, y^*)^T$
8: **end while**
9: **return** $y_{min} = \min(\boldsymbol{y})$ and the corresponding $\boldsymbol{x}_{min}$

This scheme can be summarized as follows: In the first step, an initial design with n points is evaluated and a surrogate model is fitted. The surrogate model is used for the prediction of a new design point. As long as the optimization budget is not exhausted, a new point is chosen in the parameter space based

[1] `http://www.maniactools.com/soft/midi_converter/index.shtml`.
[2] The treatment of categorical features will be discussed later.

on a so-called infill criterion derived from the surrogate model. For this point, the target function is evaluated. The surrogate model is then updated on the design extended by the new point. This model is used for the next iteration. The point with the minimal target function value is taken as the result of the optimization. In what follows, these steps will be discussed in more detail.

4.1 Initial Design and Optimization Budget

The budget should depend on the dimension of the target function (number of function parameters d) and is a tradeoff between a good optimization result and the needed time effort. The choice of the initial design should take into account that too small designs are not able to sufficiently scan the target function which might lead to local convergence. In [6], [20], and [18] the influence of the size and the structure of the initial design on the optimization result is studied.

Here, *Latin Hypercube Sampling* (LHS, [21]) designs are used for the initialization step. LHS designs are very popular in the computational optimization community due to their properties of uniformly covering the interesting parameter space and its arbitrarily selectable size (in contrast to the classical orthogonal designs). Because of the complexity of the optimization problem and the corresponding high computation time required for function evaluations, the size of the initial design is set to $5d$ and the number of sequential steps (iterations) is set to $20d$.

4.2 Surrogate Model

A surrogate model is used for proposing a new design point. Theoretically, an arbitrary regression model can be used as a surrogate model. Very popular is the so-called Kriging model [22], which can model high dimensional multimodal function landscapes in good quality already with few points.

We use the ordinary Kriging Modell [6]:

$$Y(\boldsymbol{x}) = \mu + Z(\boldsymbol{x}). \tag{5}$$

$Y(\boldsymbol{x})$ is a random variable and the error term $Z(\boldsymbol{x})$ is a Gaussian process expressing the uncertainty in $Y(\boldsymbol{x})$ having the properties: $\mathbb{E}(Z(\boldsymbol{x})) = 0, K(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = Cov(Z(\boldsymbol{x}), Z(\tilde{\boldsymbol{x}})) = \sigma_z^2 k(\boldsymbol{x}, \tilde{\boldsymbol{x}})$, where $K(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ is the so-called covariance kernel, $k(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ the spatial Covariance Function (CF), and σ_z^2 the process variance ([23]). The covariance function models the structure in the data points and is, obviously, the most decisive part of Gaussian process specification [24]. For multidimensional data, the product correlation rule is applied: $k(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \prod_{j=1}^{d} k_j(x_j, \tilde{x}_j)$.

In the last decades, many different CFs were proposed [24]. We use the 3/2 - Matérn CF:

$$k_j(x_j, \tilde{x}_j) = \left(1 + \frac{\sqrt{3}|x_j - \tilde{x}_j|}{\theta_j}\right) exp\left(-\frac{\sqrt{3}|x_j - \tilde{x}_j|}{\theta_j}\right). \tag{6}$$

The greater the distance between x_j and $\tilde{x}_j$, the smaller is the value of $k_j(x_j, \tilde{x}_j)$. Therefore, the influence of already evaluated points on the prediction of new points shrinks with increasing distance. The parameter θ_j controls the speed of influence reduction: the greater θ_j, the greater is the influence region. The CF parameters $(\theta_1, \ldots, \theta_d)^T$ as well as the process variance σ_z^2 are estimated by means of the Maximum-Likelihood method. See [6] for a detailed description.

4.3 Adapting Kriging for Categorical Parameters

One of the most important disadvantages of the standard Kriging model is its limitation to numerical influential parameters. If there are also categorical parameters to be optimized, Kriging can be extended in several manners.

A naive variant is to treat categorical parameters as if they were continuous. First, each level is assigned to an integer and in the sequential steps proposed values of the corresponding continuous parameter are rounded and converted back to the nearest categorical level.

Another possibility is the so-called dummy Kriging, where each categorical parameter with m possible values is expressed by m different parameters (called dummy variables) which take the value 1 if the corresponding value is taken and 0 otherwise. Since all these dummy variables are used in the regression model, the number of parameters in the model can be high leading to long run times for model fits.

4.4 Infill Criterion

The infill criterion is a rating function which estimates how promising a target function evaluation at a given point is. Very intuitive criteria relate to the goodness of model predictions, e.g., the lowest possible target value in the case of minimization. However, a disadvantage of such a criterion would be a possible convergence to a local optimum.

Instead, typically the Expected Improvement (EI) criterion, as proposed by [5], is used as a compromise between exploitation of the surrogate model and exploration of the function landscape. The EI criterion supports global convergence ([25]) and is the standard criterion in many applications. The EI in a point $\boldsymbol{x}$ is defined as the expected value of a positive improvement of the target function in this point:

$$\mathrm{EI}(\boldsymbol{x}) = \mathbb{E}[\max\{0, y_{min} - \hat{f}(\boldsymbol{x})_{\mathcal{D}}\}], \tag{7}$$

where y_{min} denotes the, so far, minimal value of the target function. The expected improvement should be maximized leading to the new design point $\boldsymbol{x}^* = \arg\max_{\boldsymbol{x} \in \mathcal{X}} \ \mathrm{EI}(\boldsymbol{x})$.

4.5 Optimization of Infill Criterion

In each MBO iteration we would like to choose a new point maximizing the infill criterion. To solve the corresponding non-linear optimization problem,

we use the focus search algorithm implemented in the R package **mlrMBO**[3] which successively focuses the parameter space on the most promising regions. The main idea is: Generate a random LHS design of the size N_{points} on $\mathcal{X}$ and calculate the corresponding values of the infill criterion. Then, shrink the parameter space in each dimension to the environment of the best point. Iterate the shrinking N_{maxit} times. As this procedure can lead to a local optimum, focus search should be replicated several ($N_{restarts}$) times. As our final new point $\boldsymbol{x}^*$ we will take the best point over all iterations and repetitions. We will use the following settings: $N_{points} = 10000$, $N_{maxit} = 5$ and $N_{restarts} = 3$.

5 Instance Bound Model Based Optimization

Until now we did not discuss what is meant by a function evaluation. In our application we do not just have to evaluate a target function once, but we have to evaluate multiple instances and consider, e.g., mean responses. Hence, our target function is instance bound. The instances are the music pieces on which our target function should be minimized on average. Obviously, we can apply MBO and calculate the mean F-value over all instances. However, the problem with many instances is the high computational time needed for evaluation. Therefore, we are looking for a short cut by excluding unpromising parameter settings without evaluating all instances. For simplification we will call promising settings "good" and unpromising ones "bad" in the following.

One possible approach is the instance bound SMAC method (Sequential Model-based optimization for general Algorithm Configuration) of [26], where the initialization step is only based on the evaluation of a few randomly selected instances and newly proposed points are iteratively evaluated on random instances until there is indication that the point is worse than the so far best. However, SMAC appears to be not very suitable for our purposes since it needs instance characteristics like, e.g., genre, instruments involved etc. for surrogate modeling, which is not possible for online applications.

Therefore, in [7] we proposed another method called FMBO (Fast Model Based Optimization) for onset detection. This method, however, can be used in applications with multiple instances. The following scheme comprises the FMBO method, where M_{sur} is the surrogate model introduced in Section 4.2. The idea is to minimize the negative F-measure. In FMBO we think of $y = f(\boldsymbol{x})$ as the mean negative F-measure over all instances in point $\boldsymbol{x}$.

FMBO is based on the observation that "bad" parameter settings of the OD algorithm are reflected in smaller F-values in most instances. Therefore, such bad settings can already be detected after few iterations. In these cases, the other instances should be waived for the estimation of the mean F-value.

Let k_{all} be the overall number of instances and $k_{pretest}$ the number of instances used for the pretest applied for the distinction between "good" and "bad" parameter settings. In order to select the $k_{pretest}$ instances, we extend the clustering based method proposed in [7]. We not only store the F-values but

[3] Developing version. `https://github.com/berndbischl/mlrMBO`.

also the D-values of each point in the initial design for all k_{all} instances. Then, we cluster the k_{all} instances based on their F- and D-values into $k_{pretest}$ clusters by means of the *k-means* algorithm ([27]). Randomly selected representatives of the clusters, one for each cluster, build the pretest data base (lines 2 - 4 of the algorithm). In this way, we strive to reach the greatest diversity of the music pieces in this data base. Afterwards, we choose the *pretest model* $M_{pretest}$ by linearly regressing the mean F-value over the k_{all} instances to the individual F-values of the $k_{pretest}$ instances (line 5).

1: generate an initial design $\mathcal{D} \subset \mathcal{X}$
2: evaluate $\mathcal{D}$ on all instances: $\boldsymbol{y} = f(\mathcal{D})$
3: cluster all instances in $k_{pretest}$ clusters
4: choose random representatives for the selection set
5: fit $M_{pretest}$: $y \sim$ individual F-values of selected instances
6: **while** budget is not exceeded **do**
7: fit surrogate model M_{sur} on $\mathcal{D}$
8: find $\boldsymbol{x}^*$ by infill criterion optimization
9: evaluate $\boldsymbol{x}^*$ on the cluster representatives
10: predict mean F-measure: $\hat{y}^* = M_{pretest}(\boldsymbol{x}^*)$
11: calculate 99% confidence interval: $[\hat{y}_u^*, \hat{y}_o^*]$
12: **if** $\hat{y}_u^* \leq y_{min}$ **then**
13: evaluate $\boldsymbol{x}^*$ on all k_{all} instances: $y^* = f(\boldsymbol{x}^*)$
14: update $\mathcal{D}$ and $\boldsymbol{y}$: $\mathcal{D} \leftarrow (\mathcal{D}, \boldsymbol{x}^*)^T$ and $\boldsymbol{y} \leftarrow (\boldsymbol{y}, y^*)^T$
15: update model $M_{pretest}$ using new observation
16: update elements of $\boldsymbol{y}$ corresponding to "bad" points
17: **end if**
18: update $\mathcal{D}$ and $\boldsymbol{y}$: $\mathcal{D} \leftarrow (\mathcal{D}, \boldsymbol{x}^*)^T$ and $\boldsymbol{y} \leftarrow (\boldsymbol{y}, \hat{y}^*)$
19: **end while**
20: **return** $y_{min} = \min(\boldsymbol{y})$ and corresponding $\boldsymbol{x}_{min}$.

In the following repeated steps (while loop), the proposed points are first evaluated on the instances of the pretest data base (line 9) and classified as "good" or "bad". This can be done in different ways. We base our decision on prediction intervals: for the point $\boldsymbol{x}^*$ we calculate the prediction $\hat{y}^*$ and the 99% prediction interval $[\hat{y}_u^*, \hat{y}_o^*]$ based on $M_{pretest}$ (lines 10 - 11). If the lower limit of the prediction interval is smaller than the so far reached minimum of the averaged F-values, then $\boldsymbol{x}^*$ is classified as a "good" point, otherwise as "bad".

The "good" points are evaluated at the remaining $k_{all} - k_{pretest}$ music pieces of the complete data base (line 13), $\mathcal{D}$ and $\boldsymbol{y}$ are updated (line 14), and the pretest model is newly estimated in order to include the new information in the additional observations (line 15). If a "bad" point is found, $y^* = f(\boldsymbol{x}^*)$ is estimated only by $\hat{y}^*$ and given back to the MBO loop (line 18). It is important to notice that these estimations are updated after each new update of the pretest model (line 16).

For both the clustering algorithm and the choice of the pretest model there would be many sensible alternatives which could be compared in further studies. For example, also the application of a classification method to divide "good" and "bad" points would be intuitive.

6 Experiments

In this paper, we compare eight optimization techniques for the onset detection algorithm, namely MBO_naivKM, FMBO_naivKM, MBO_dummyKM, and FMBO_dummyKM applied for online and offline OD. This corresponds to MBO and FMBO algorithms with naive and dummy Kriging surrogate models (KM) in the case where only past observations of music pieces are available (online case) and in the case where a certain interval of future signal is allowed (offline case).

The most important questions to the experiments are:

- Is there a relevant performance loss when using online onset detection compared to offline?
- Is dummy Kriging better than naive Kriging?
- Is the proposed FMBO competitive with MBO?

The size of the initial design and the number of iterations are 70 ($5d$) and 280 ($20d$) points, respectively, where $d = 14$ is the number of parameters in our study (see Section 2). All further MBO and FMBO settings were discussed in the previous section. In order to show the efficiency of MBO, a random search with 350 points (generated via an LHS design) is also conducted both for online and offline onset detection.

6.1 Validation of Optimization Approaches

A very important aspect when comparing many optimization strategies is dealing with over-fitting to the training data. In order to avoid a very good fit on the training data and a much worse performance on new test data, we use the resampling technique ([28]). More specifically, we apply k-fold cross validation to parameter setting evaluation. The main idea of k-fold cross-validation is presented in Figure 1: Split the data in k disjunct blocks of equal size, conduct the optimization procedure on the training data of $k - 1$ blocks and validate the best found parameter setting on the remaining block (test data). Every data block is used once as a test set where the corresponding cross-validated F-values (averaged over the associated music pieces) are computed and analyzed in the following. The whole data set consists here of 460 music pieces so that each block contains 46 pieces when $k = 10$. $k_{pretest}$ is chosen to be 5% of the training data set.

As MBO is a stochastic approach which is usually affected by the distribution of the initial design points, each optimization strategy is replicated five times in

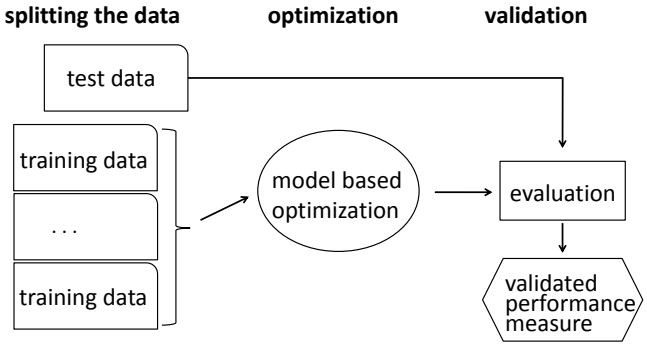


Figure 1: Cross-validation of optimization strategies.

each cross validation step. This results in 50 runs for each strategy (including random search). Such procedure is also called 5 times replicated 10-fold cross-validation. Note, that in one replication the same initial design is used for all strategies in order to ensure uniform starting conditions. On the used computer system one optimization run of MBO_naiveKM and MBO_dummyKM requires on average 79 and 84 hours, respectively[4] MBO optimization is conducted using developing version of **mlrMBO** R package. The OD algorithm as well as the FMBO extension are implemented in the *R* programming language ([30]) and can be provided on request.

6.2 Comparison of Optimization Performance

A difficult problem is how to compare MBO and FMBO. FMBO can be seen as an approximation of MBO which on the one hand is probably a bit worse but on the other hand much faster. There are two possible points of view to compare the two approaches. One possibility is based on the same number of iterations and the other on the same number of instance evaluations. Which approach is better depends on the runtime costs of the instance evaluations versus the runtime costs of fitting the Kriging model. Taking the below time analysis into account, the more sophisticated comparison seems to be based on the number of instance evaluations. However, to achieve a better generalization to other optimization tasks, we will consider both points of views.

Figure 2 illustrates the distribution of the cross-validated F-values for all optimization strategies for the same number of iterations. We observe similar ranges of F-values as in previous studies ([3, 17]) which just handle subsets of our complex data base.

Additionally, pairwise Wilcoxon signed rank tests ([31]) were conducted to answer the three research questions mentioned above. The two-tailed Wilcoxon

[4]The experiments were executed in parallel using the **BatchExperiments** R package ([29]) on the Linux-HPC cluster system (`http://lidong.itmc.tu-dortmund.de/ldw/index.php?title=System_overview&oldid=259`) of TU Dortmund University.

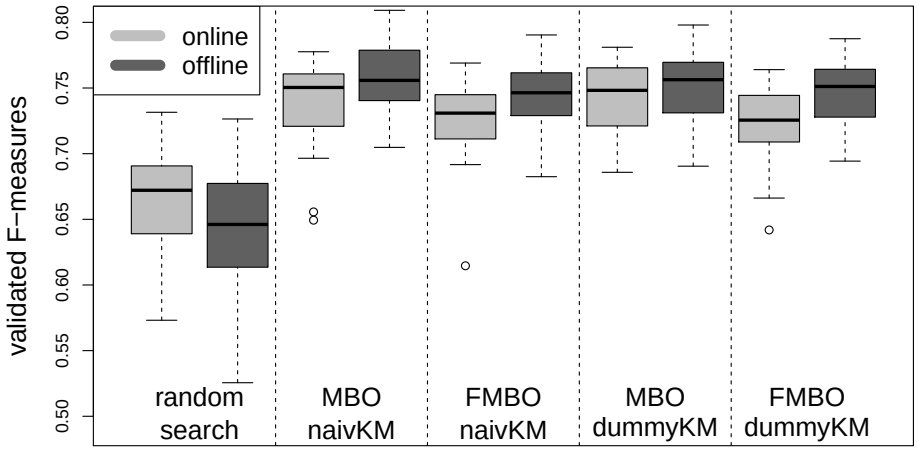


Figure 2: Cross-validated results of the optimization strategies.

test was applied instead of the t test as the data appears to be not normally distributed (according to q-q-plots). Due to the fact that many tests are applied on the same data, multiple test level adaption was applied according to the Bonferroni-Holm correction ([32]).

The tests show that offline onset detection provides significantly better F-values than the online version and that MBO is significantly better than FMBO[5]. However, there is no significant difference between dummy Kriging and naive Kriging. This is a positive result as, firstly, naive Kriging is faster than the statistically correct dummy Kriging model and secondly, many infill criteria and the corresponding optimization algorithms can only handle continuous parameters.

The significantly better results of the offline approach are obviously due to additional information about the future signal behavior. However, the performance difference appears to be acceptable in view of the benefits of the online approach (e.g. for hearing aids applications). Note that the online approach surprisingly leads to better F-values for the random search.

The third and most interesting research question requires a more detailed analysis. Although the FMBO results are significantly worse than those of MBO (except for offline dummy Kriging), the comparison above did not consider the massive runtime saving of FMBO.

We will not analyze time savings in seconds, though, as the computational time of OD algorithms depends on its parameter settings (mainly on N and h) and the length of music pieces, but we will calculate savings in music piece evaluations when applying FMBO instead of MBO (Figure 3).

The astonishingly high saving of 80% in the median for FMBO_naivKM and 82% for FMBO_dummyKM can be explained by the fact that on average only

[5]Note that the sample sizes of 50 observations are relatively large so that already small differences in the sampling distributions can be found as significant even if they appear to be not relevant.

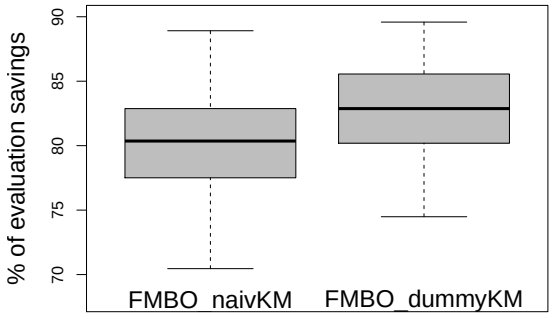


Figure 3: Percentage of music piece evaluations saved by FMBO compared to MBO.

in 41 and 33 of the 280 steps, respectively, the proposed points were selected as "good" points. In the remaining "bad" steps just 25 music pieces of the pretest data base were evaluated.

A detailed time analysis of the conducted experiments shows that on average for MBO_naivKM music piece evaluations consume 92% of the entire time while 2% and 6% is used for fitting the naive Kriging model and optimizing the EI infill criterion, respectively. The analogue time proportions for MBO_dummyKM are 88% - 4% - 8%. On average, fitting of a dummy Kriging model needs a factor of 2.13 more time than naive Kriging. The relatively high time percentage for EI optimization can be explained by the extensive setting of the focus search algorithm. The time for $M_{pretest}$ fitting can be neglected due to very low percentages.

After we compared MBO vs. FMBO based on the same number of iterations (Figure 2), let us now compare them based on the same number of music piece evaluations. For that reason, the number of instance evaluations made by an FMBO application is divided by the size of the training data set and rounded up. In this manner, the number of equivalent complete data set evaluations is calculated. Afterwards, the optimization path of the associated MBO run (in the same validation step and with the same initial design) is cut to this number of iterations and its best already achieved F-value is saved. This approach is abbreviated by MBO_cut.

In Figure 4, the comparison of FMBO and MBO_cut shows a significant superiority of the FMBO approach for both surrogate models. Although the number of music piece evaluations is nearly the same, the computational time for FMBO might be slightly larger due to modeling overhead. This overhead can be ignored if the computational time for music piece evaluations is noticeable larger than the $M_{pretest}$ and M_{sur} model fitting time.

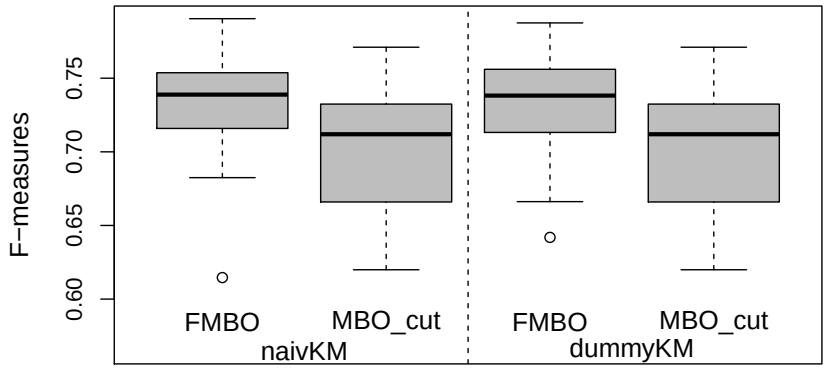


Figure 4: F-values of FMBO compared to MBO_cut.

6.3 Best Settings of OD Algorithm

Online Onset Detection The best parameter settings for online detection achieving an F-value of 0.78 are: $N = 1024$, $h = 530$, *window.fun* = *Hamming*, $\alpha = 0.72$, $m = 0.37$, $r = 30.91 \cdot 10^5$, *odf.fun* = *SF*, *moving.fun* = *median*, $\lambda = 1.68$, $t(r_T) = 0.008$, $t(l_T) = 0.24$, $t(r_O) = 0.007$, $t(l_O) = 0.10$ and $t(min.dist) = 0.049$. Figure 5 shows the distribution of some selected OD parameters of the best found settings over the 200 runs of all MBO and FMBO methods (i.e., for naivKM and dummyKM) for the online approach, where the settings using $odf.fun = SE$, being the most often chosen ODF, are marked dark.

Analyzing Figure 5 and the distribution of the remaining OD parameters (not shown here), we can state that the most frequently chosen N is 1024 samples and rather high h are preferred (almost equal to N). Also, *Hamming* and *Gauss* window functions appear to be meaningful choices. The most successful *odf.fun* is spectral flux, but also, among others, complex domain and rectified complex domain were selected.

Adaptive whitening seems to have no effect as the best chosen parameter values vary almost uniformly in the permitted intervals. To verify this presumption, we generated 10 random combinations of m and r parameters in addition to the $[0, 0]$ (total whiting), $[0, 10^7]$ and $[1, 10^7]$ settings and computed the F-values fixing all further parameters to the best settings mentioned above. As expected, the F-value for total whitening is 0, while all other F-values are nearly the same. For this reason we propose not to use this pre-processing option for time saving purposes.

The smoothing parameter α should be set at least to 0.6 while even higher values are preferred (i.e. weak smoothing). Regarding the *mov.fun* for thresholding, the *mean* option appears to be advisable with λ in the interval [1.4, 1.8].

While the distribution of $t(l_T)$ shows a light negative skew with the maximum at 0.5 s and the minimum at about 0.1 s (so that rather large time intervals

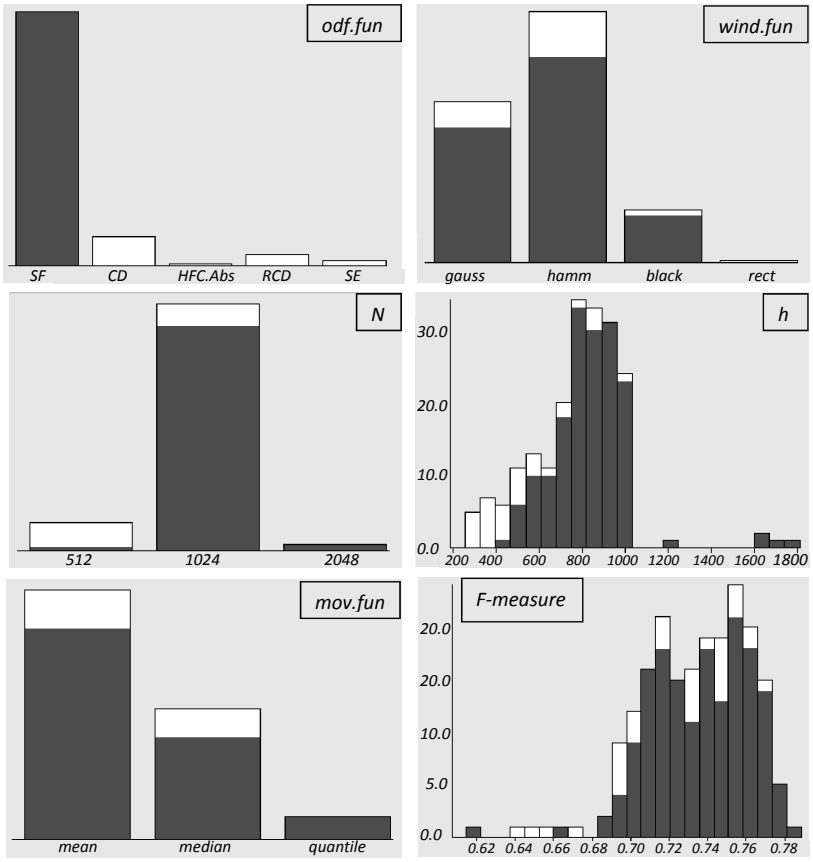


Figure 5: Distributions of optimal settings for 5 selected parameters and the F-value distribution. Dark areas correspond to settings with spectral flux.

are preferred for thresholding), the distribution of $t(l_O)$ is symmetrical around 0.1 s in the interval [0.05, 0.15] s (larger intervals are hence not needed for maximum localization). As the maximally allowed future time period is 10 msec (corresponding to 441 samples for frequently chosen N and h settings), we can conclude that practically no future information is involved and latency time can by assumed to be 0. Finally, $t(min.dist)$ is chosen near to 0.05 s in almost all cases.

Offline Onset Detection The best parameter settings for offline detection achieving an F-value of 0.81 are: $N = 2048$, $h = 1242$, $window.fun = Blackmann$, $\alpha = 0.75$, $m = 0.95$, $r = 7591$, $odf.fun = SF$, $moving.fun = median$, $\lambda = 1.27$, $t(r_T) = 0.23$, $t(l_T) = 0.45$, $t(r_O) = 0.06$, $t(l_O) = 0.09$ and $t(min.dist) = 0.042$.

For offline OD just two settings of N are selected: 2048 (70%) and 1024 (30%), while the most h values are a bit above 50% of N. Although the two best settings apply to a *Blackmann* window, the *Hamming* and *Gauss* windows are preferred more frequently. For adaptive whitening the same conclusion can be drawn as in the online case. $t(r_T)$ is relatively uniformly distributed in the whole permitted interval, while $t(l_T)$ is again concentrated on the longer times (between 0.3 and 0.5 s). The distributions of $t(l_O)$ and $t(r_O)$ are unimodal and symmetrical in the intervals [0.05, 0.15] s and [0.03, 0.12] s, respectively. Finally, $t(min.dist)$ is now uniformly distributed in [0, 0.05] s.

7 Conclusion

In this paper, we particularly define an online onset detection algorithm with three categorical and nine continuous parameters. In contrast to most other state-of-the-art papers, the whole set of parameters is optimized simultaneously in order to consider all possible interactions, for both online and offline OD.

Optimizing such complex algorithms is not a trivial task. For solving we use sophisticated model based optimization techniques: sequentially evaluating the promising parameter settings according to a surrogate model and an appropriate infill criterion. As the popular Kriging model is limited only to continuous parameters, we compare two extensions for categorical parameters: naively handling them as continuous and building dummy variables.

In view of the large evaluation time on the whole data base we implement an instance bound fast MBO where in each iteration a small pretest data set classifies the proposed settings in promising and unpromising ones. The complete evaluation is continued just in promising points, otherwise the performance is estimated by the pretest model.

As the optimization results depend on the training data, we first construct a large and heterogeneous data base including many manually labeled music pieces frequently used in the literature as well pieces generated from MIDI templates (using synthetic and real tones). Secondly, a 5 times replicated 10-fold validation is applied for avoiding a possible over-fitting.

The experiments show that the offline OD performs significantly better than online OD as it uses additional information about the future signal behaviour. The loss in the F-values is on average 0.016 indicating an acceptable performance loss compared to the advantages for online applications. Regarding the surrogate model, the naive Kriging approach is as successful as the statistically more correct dummy alternative which is also more time consuming.

On the one hand, when considering the same number of iterations MBO performs significantly better than FMBO, the average loss in F-values being merely 0.007. Compared to MBO, FMBO needs just about 20% of music piece evaluations (which is by far the most time consuming part) and enables optimization in acceptable time. On the other hand, however, when considering the same number of music piece evaluations, FMBO shows significantly better results than MBO, the gain in the F-values being on average 0.03.

Finally, the best parameter settings for both online and offline OD were discussed. For both variants spectral flux is the best feature. In the offline case larger window sizes are more preferable than in the online case. Interestingly, the adaptive whitening pre-proceeding step appears to not affect the F-values in our algorithm. A possible reason for this finding might be the robustness of the threshold function against different scales of the OD functions.

In our future work, we aim, on the one hand, to define and optimize a multivariate onset detection algorithm were – instead of only one – several OD features are considered simultaneously and a classification model is included for identification of the signal windows corresponding to the tone onsets. On the other hand, we intend to improve and extend the FMBO in several respects.

Acknowledgment

This paper is based on investigations of the projects C2 and B3 of SFB 823, which are kindly supported by the German Research Foundation (DFG).

References

[1] J. P. Bello, L. Daudet, S. A. Abdallah, C. Duxbury, M. E. Davies, and M. B. Sandler, "A tutorial on onset detection in music signals," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5-2, pp. 1035–1047, 2005.

[2] S. Dixon, "Onset detection revisited," in *Proceedings of the 9th International Conference on Digital Audio Effects (DAFx'09)*, 2006, pp. 133–137.

[3] A. Holzapfel, Y. Stylianou, A. Gedik, and B. Bozkurt, "Three dimensions of pitched instrument onset detection," *IEEE Transactions on Audio, Speech, and Language Ppocessing*, vol. 18 (6), pp. 1517–1527, 2010.

[4] C. Rosão, R. Ribeiro, and D. Martins De Matos, "Influence of peak selection methods on onset detection," in *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR'12)*, 2012, pp. 517–522.

[5] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[6] V. Picheny, T. Wagner, and D. Ginsbourger, "A benchmark of kriging-based infill criteria for noisy optimization," *Structural and Multidisciplinary Optimization*, vol. 48, no. 3, pp. 607–626, 2013.

[7] N. Bauer, K. Friedrichs, and B. B. C. Weihs, "Fast model based optimization of tone onset detection by instance sampling," in *Proceedings of the*

European Conference on Data Analysis, Accepted, Ed. Springer International Publishing, 2015.

[8] F. J. Harris, "On the use of windows for harmonic analysis with the discrete fourier transform," *IEEE*, vol. 66 (1), pp. 51–83, 1978.

[9] D. Stowell and M. Plumbley, "Adaptive whitening for improved real-time audio onset detection," in *Proceedings of the International Computer Music Conference (ICMC'07)*, 2007, pp. 312–319.

[10] E. Benetos and S. Dixon, "Polyphonic music transcription using note onset and offset detection," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 37–40.

[11] N. Bauer, J. Schiffner, and C. Weihs, "Einfluss der musikinstrumente auf die güte der einsatzzeiterkennung," Tech. Rep. 10/12, 2012.

[12] W. A. Schloss, "On the automatic transcription of percussive music - from acoustic signal to high-level analysis," Master's thesis, Stanford University, 1985.

[13] P. Masri, "Computer modeling of sound for transformation and synthesis of musical signal," Ph.D. dissertation, University of Bristol, 1996.

[14] G. Peeters and X. Rodet, "A large set of audio feature for sound description (similarity and classification) in the cuidado project," Ircam, Tech. Rep., 2004.

[15] N. Bauer, K. Friedrichs, D. Kirchhoff, J. Schiffner, and C. Weihs, "Tone onset detection using an auditory model," in *Data Analysis, Machine Learning and Knowledge Discovery*, M. Spiliopoulou, L. Schmidt-Thieme, and R. Janning, Eds. Springer International Publishing, 2014, pp. 315–324.

[16] F. Eyben, S. Böck, B. Schuller, and A. Graves, "Universal onset detection with bidirectional long short-term memory neural networks," in *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR'10)*, J. S. Downie and R. C. Veltkamp, Eds. International Society for Music Information Retrieval, 2010, pp. 589–594.

[17] S. Böck, F. Krebs, and M. Schedl, "Evaluating the online capabilities of onset detection methods," in *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR'12)*, 2012.

[18] N. Bauer, J. Schiffner, and C. Weihs, "Comparison of parameter optimization techniques for a music tone onset detection algorithm," in *Proceedings of the 4th Meeting on Statistics and Data Mining (MSDM)*, 2013, pp. 28–34. [Online]. Available: http://www.tasa-online.com/

[19] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "Rwc music database: Music genre database and musical instrument sound database." in *Proceedings of ISMIR'03*. International Society for Music Information Retrieval, 2003, pp. 229–230.

[20] N. Bauer, J. Schiffner, and C. Weihs, "Comparison of classical and sequential design of experiments in note onset detection," in *Data Analysis, Machine Learning and Knowledge Discovery*, B. Lausen, D. V. den Poel, and A. Ultsch, Eds. Springer International Publishing, 2013, pp. 501–509.

[21] M. D. McKay, R. J. Beckman, and W. J. Conover, "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 55–61, 1979.

[22] D. G. Krige, "A statistical approach to some basic mine valuation problems on the witwatersrand," *Metallurgical and Mining Society of South Africa*, vol. 52, no. 6, pp. 119–139, 1951.

[23] M. Schonlau, "Computer experiments and global optimization," Ph.D. dissertation, University of Waterloo, 1997.

[24] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. The MIT Press, 2006.

[25] D. R. Jones, "A taxonomy of global optimization methods based on response surfaces," *Journal of Global Optimization*, vol. 21, no. 4, pp. 345–383, 2001.

[26] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Learning and Intelligent Optimization*, ser. Lecture Notes in Computer Science, C. A. C. Coello, Ed. Springer, 2011, vol. 6683, pp. 507–523.

[27] W. M. A. Hartigan, J. A., "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society, Series C (Applied Statistics)*, vol. 28, pp. 100–108, 1979.

[28] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York: Springer New York Inc., 2001.

[29] B. Bischl, M. Lang, and O. Mersmann, *BatchExperiments: Statistical experiments on batch computing clusters.*, 2014, r package version 1.2. [Online]. Available: http://CRAN.R-project.org/package=BatchExperiments

[30] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2014. [Online]. Available: http://www.R-project.org/

[31] F. Wicoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[32] M. Aickin and H. Gensler, "Adjusting for multiple testing when reporting research results: the bonferroni vs holm methods," *American Journal of Public Health*, vol. 86, pp. 726–728, 1996.