DICTIONARY LEARNING ASSISTED TEMPLATE MATCHING FOR AUDIO EVENT DETECTION (LEGATO)

Aggelos Pikrakis

Dept. of Informatics
University of Piraeus, Greece
pikrakis@unipi.gr

Yannis Kopsinis

Libra MLI
Edinburgh, UK
kopsinis@ieee.org

ABSTRACT

We submit a two-stage scheme for the detection of audio events in synthetic audio.At a first stage, the endpoints of candidate events are located by means of an unsupervised method based on dictionary learning. At a second stage, each candidate event is matched against all provided event templates using a variant of the Smith-Waterman algorithm. This stage includes a hypothesis test against scores generated by random permutations of the feature sequences corresponding to the candidate event and each reference template. The unknown event is classified to the reference template that generates the highest computed score. The segment-based values of the F-measure and Error Rate, when the method is tested on the provided development dataset of the 2016 DCASE Challenge, are 64.01% and 50.75%, respectively. The corresponding values during event-based evaluation are 62.52% and 51.85%.

***Index Terms*—** Dictionary learning, template matching, Smith-Waterman

1. METHOD DESCRIPTION

The proposed method addresses the 2nd task of the DCASE2016 challenge (sound event detection in synthetic audio) in two stages: a segmentation stage and a classification stage.

1.1. Segmentation

The segmenter is based on recent work by the authors in [1], which was developed in the context of singing voice detection and employs dictionary learning. More specifically, the audio recording to be segmented is first represented by a sequence of feature vectors. The adopted feature extraction scheme uses a moving window technique to compute the spectrogram of the audio signal. The length of the moving window is 46.4 ms and the hop size is 11.6 ms. The spectrogram serves to compute a bark-band representation of the signal by summing together the short-time Fourier transform coefficients that reside within the limits of the respective bark band, thus yielding a sequence of 24-dimensional vectors.

The KSVD algorithm [2] is subsequently used to learn a dictionary of 64 atoms which are combined to yield a 3-sparse representation of the feature sequence. The frequency of excitation of the dictionary atoms during the reconstruction procedure is used to estimate their probability of appearance, based on which the information content of each atom is computed. Subsequently, the information content of each feature vector is computed by summing the information content of the dictionary atoms that participate in its reconstruction. The feature vectors are then classified as background vectors if their information content falls below an automatically derived threshold. For more information, the reader is referred to [1].

At a next step, all the vectors that were classified as background, are used to learn a new dictionary of 64 atoms (3-sparse representation again). This new dictionary is then used to reconstruct all the feature vectors of the audio recording. It is expected that the reconstruction error will increase significantly whenever non-background vectors are encountered and fall rapidly whenever the background signal is reconstructed. Due to the stochastic initialisation of the KSVD algorithm and the inherent difficulty in computing a global threshold that will decide which frames belong to the background, the whole procedure has to be repeated several times (20 times in our system) and the average reconstruction error is computed for each frame. Let $r = \{r_i, i = 1, \ldots, M\}$ be the resulting sequence of reconstruction errors, where M is the length of the feature sequence. An example is shown in Figure 1.

Subsequently, an onset detector is applied on sequence r. The onset detector is a sliding window, 0.1 s long (9 frames in our system implementation), which is centred at each frame and serves as a detector of abrupt changes. Assuming that the onset detector is centred at the n-th frame, the sliding window covers frames

$$x(n-4), x(n-3), \ldots, x(n), x(n+1), \ldots, x(n+4)$$

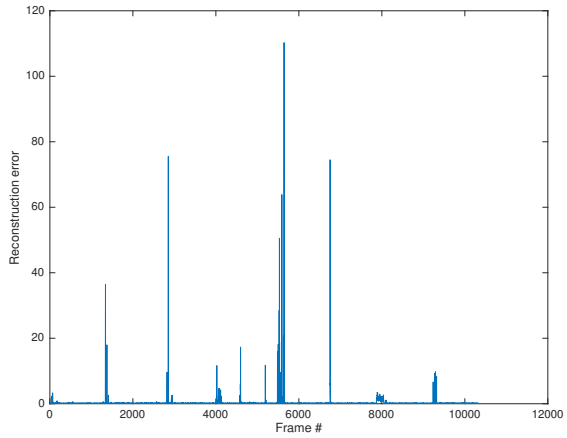

Figure 1: Sequence of reconstruction errors for an audio recording of the development dataset.

The quantity

$$s(n) = \sum_{k=1}^{4} x(n+k) - \sum_{k=1}^{4} x(n-k)$$

is then computed. Increasing values of $s(n)$ indicate that the sliding window enters an audio event. Accordingly, decreasing values signal the offset of the event. Therefore zero crossings are expected to be encountered in the middle of events. An example of sequence s is shown in Figure 2.

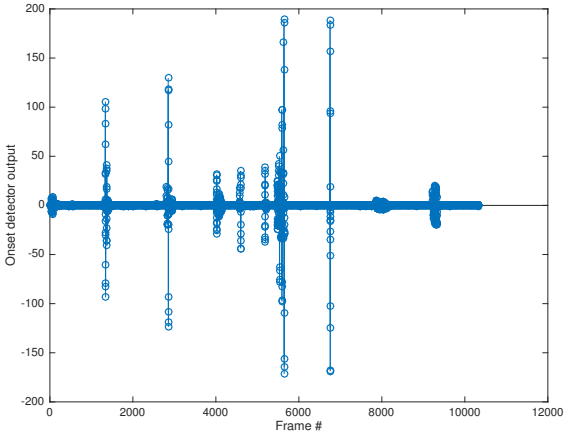

Figure 2: Sequence at the output of the onset detector when the input is the error sequence of Figure 1.

Sequence s is noisy and needs to be filtered. To denoise it, we use clipping. The clipping threshold is computed as follows: the maxima of $\mid s \mid$ are detected and are sorted in descending order. The resulting sorted sequence (Figure 3) exhibits a "knee". The location of the "knee" is automatically detected using a standard slope-based method and the value at the "knee" is clipping threshold for s. Let c be the clipped version of s. Figure 4 shows the clipping threshold superimposed on s.

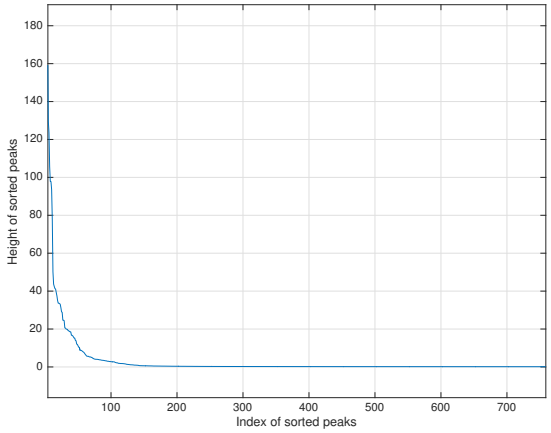

Figure 3: Height of sorted peaks of sequence s. A "knee" can be observed.

The clipped sequence c is then scanned from left to right to detect segments consisting mainly of non-zero values. Here, the term "mainly" refers to the fact that we permit small gaps of zeros to be part of a segment. In our system, such gaps can be at most 0.2 s long. This is because a lot of audio events exhibit such signal discontinuities at the frame level.

Finally, segments shorter than 60 ms are discarded. In addition, any segment that does not contain at least 6 frames whose clipped reconstruction error exceeds three times the computed threshold is discarded. This last post-processing stage aims at discarding segments that survived the clipping threshold marginally. This is why a higher threshold is imposed this time. If this higher threshold is applied at the first place, unwanted over segmentation will occur due to formed gaps. This is why it is preferable to start with a more conservative threshold, let segments with small gaps be formed and then apply the final higher threshold to filter out results.

1.2. Classifier

The output of the segmenter is a sequence of segments, i.e., a sequence of pairs of endpoints. Let T_i be the subsequence of feature vectors of the initial recording that lie between a pair of endpoints. T_i is treated as an unknown template that will be matched (aligned) against the bark band representations of all reference templates, $P_j, j = 1, \ldots, 240$ of the training set. The adopted alignment algorithm is the well-known

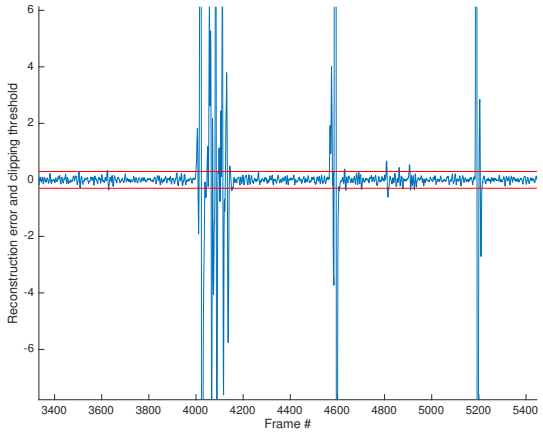

Figure 4: Onset function, s, and clipping threshold. For the sake of clarity of presentation only part of the sequence is shown on both axes.

Smith-Waterman algorithm [3]. We use the cosine of the angle of two feature vectors as a local similarity measure. To speed up computations, the search range for node predecessors horizontally and vertically is limited to 5 frames. The gap penalty is set equal to $1/3$ after experimentation. Each alignment returns a similarity score, S_{ij}, $i = 1, \ldots, L_s$ and $j = 1, \ldots, 240$, where L_s is the number of segments at the output of the segmenter. This idea was investigated by the authors in [4] in the context of event detection in soundtracks from movies.

To test if S_{ij} is a significant result, each P_j is shuffled 30 times and each random permutation is aligned against T_i. The mean score over these runs is then computed. This procedure is repeated by producing 30 random permutations of T_i and aligning each permutation with P_j. A second mean score is computed and the highest of the two mean scores is considered to be a good approximation of a random alignment score, n_{ij}, which is subsequently subtracted from S_{ij} to yield the final matching score, $S_{ij} - n_{ij}$, of T_i against P_j. After this score is computed for all 240 templates, T_i is classified to the class of the highest score.

2. PERFORMANCE EVALUATION

The method was tested on the development dataset of the challenge that consists of 18, 2 min long uninterrupted audio recordings. The performance measures were computed based on the script provided by the organisers and are listed in Table 1. It follows that the segment and event based values of the F-measure of the proposed method outperform the baseline algorithm by 23.5% and 32.2%, respectively. Similarly, the Error Rates drops by 27.8 percentage units in the segment-based case and is equal to 51.85% in the event based evaluation.

Table 1: Performance of the proposed method (baseline results in parentheses).

	Segment-based	Event-based
Pre:	0.7570	0.7385
Rec:	0.5545	0.5421
F:	**0.6401** (0.416)	**0.6252** (0.303)
ER:	**0.5075** (0.7859)	**0.5185**
S:	0.1161	0.1313
D:	0.3294	0.3266
I:	0.0620	0.0606

3. CONCLUSIONS

We submitted and presented a two-stage method for sound event detection in synthetic audio which does not rely on supervised machine learning techniques and separates the segmenter from the classifier. The segmenter is based on unsupervised learning and the classifier is a template matching scheme that operates on sequences of multi-dimensional feature vectors and uses random permutations of the feature sequences to estimate the significance of the returned results. Compared to the provided baseline method, the proposed scheme exhibits promising performance.

4. REFERENCES

[1] A. Pikrakis, Y. Kopsinis, N. Kroher, and J.-M. Diaz-Banez, "Unsupervised singing voice detection using dictionary learning," in *Proceedings of EUSIPCO, Budapest, Hungary*, 2016 (to appear).

[2] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[3] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.

[4] M. Psarakis, A. Pikrakis, and G. Dendrinos, "Fpga-based acceleration for tracking audio effects in movies," in *Proceedings of the 20th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2012, pp. 85–92.