

ONSET DETECTION WITH ARTIFICIAL NEURAL NETWORKS FOR MIREX 2005

Alexandre Lacoste
University of Montreal
Montreal, Quebec
H3T 1J4 Canada
lacostea@iro.umontreal.ca

Douglas Eck
University of Montreal
Montreal, Quebec
H3T 1J4 Canada
douglas.eck@umontreal.ca

Keywords: Onset Detection, Neural Networks, Machine Learning.

ABSTRACT

In this document, we describe two onset detection algorithms that have participated to the Mirex 2005 onset detection contest. The proposed algorithms, first classify frames of a spectrogram into onset or non-onset, using a feedforward neural network. From the classification of each frames, it extract the onset times, using a simple peak picking algorithm, based on a moving average.

1 INTRODUCTION

Onset detection is a rather simple task and is much useful for a wide range of applications. It is therefore interesting to develop a fast and robust onset detector for a wide range of music. Such an onset detector would be useful for:

- Tempo extraction: Interonset histogram gives a probability distribution for all tempos.
- Music editing: Onsets break the file into logical parts.
- Automatic music transcription: Onsets point to the beginning of notes.
- Music classification: Onset location can help for merging frame features.
- Music fingerprinting: List of onset times can serve as a robust fingerprint for songs.

Most onsets detection algorithms in the literature are based on energy variation, either in the time domain or in the time-frequency domain. This method is useful for percussive instruments but gives a bigger error rate on instruments that have a smoother time envelope. Some algorithm take advantage of the information provided in the phase part of a time-frequency domain [Bello and Sandler (2003)].

Generally, all those algorithms can be separated into 3 main steps

1. Use a known transform to represent the data in a more useful domain. The most popular, is a spectrogram.

2. Use a more specific transform, to make the onsets more salient. It might be a filter that detects positive slopes.
3. Finally use a peak picking algorithm to select the onsets.

There is not much freedom in the last step, we can modify a threshold to accept or reject the peaks. For the first step, there are few interesting transforms. Therefore our algorithm mainly focuses on the second step. It uses neural networks to find the most appropriate transform.

2 ALGORITHM DESCRIPTION

This algorithm uses neural networks to classify frames into two classes: onset or non-onset. The structure of the algorithm is in 3 steps as described earlier. See figure 1. A second algorithm have been developed which combine n time the first algorithm trained with different hyperparameters. It also uses the tempo information for a better onset selection. The outputs of all algorithms and all tempo traces are combined to make some sort of feature spectrogram and finally the first algorithm is applied back on this spectrogram. See figure 2.

The first algorithm is called single-net and the second algorithm is called many-nets.

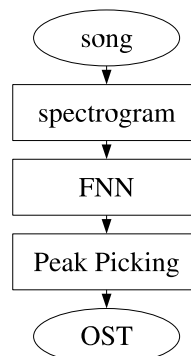


Figure 1: Single-net, flow chart. It computes the spectrogram, uses a neural network to find an onset trace and it uses peak picking to find the onset times.

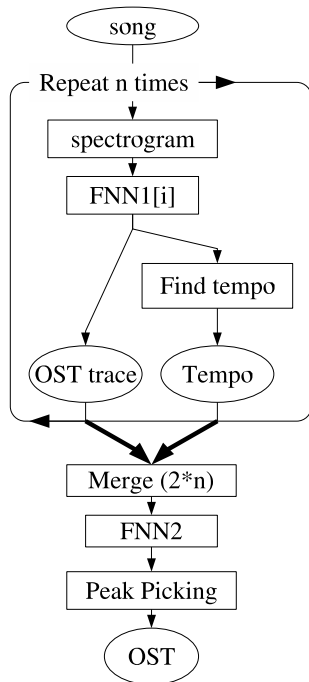


Figure 2: Many-nets, flow chart. It is only repeating the single-net algorithm n times with different hyperparameters and then uses another neural network to combine everything. It also uses the tempo to tell if the actual time is part of the tempo

2.1 Time-frequency domain transform

The information available for the classification is mainly local. Some global information like the tempo, might be useful but only if the song is rhythmic. To catch the local information of the onset we need a window length of around 100 ms. For a sampling rate of 22050 Hz, this means 2205 variables which leads to too many parameters if directly connected to the net. We therefore need a suitable representation for some dimensionality reduction.

For our algorithm we have used time-frequency transform, using either short time Fourier transform (STFT) and constant-q transform [Brown (1991)]. We've used both the magnitude and the phase part of the transform. The phase plane was transformed as cited in [Bello and Sandler (2003)]. For the magnitude plane, we took the logarithm of the amplitude for low sound detection.

The neural network was used for combining those planes, but the phase planes didn't provide better results. They seemed to be helpful only for trivial onsets. Therefore, we only used the logarithm of the magnitude of both linear and logarithmic frequency scale spectrogram. Combining linear and logarithmic spectrogram was useful. Even if they were redundant, they provided different resolutions. Typical resolution for spectrogram were 200 bin/s for good time precision and 150 frequency bins ranging from 50 Hz to 10 KHz.

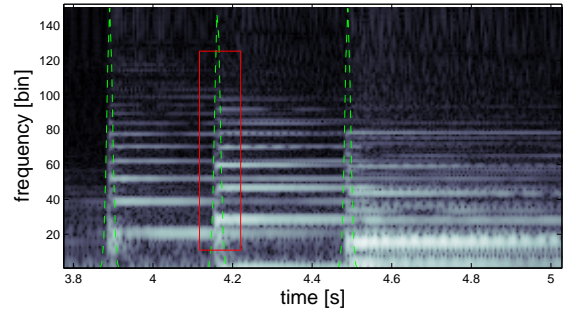


Figure 3: Constant-Q transform of a piano song with labeled onsets. The green dashed line is the onset trace, It correspond to the ideal input to the peak picking algorithm. The red box is a window of the neural network, for a particular time and a particular frequency.

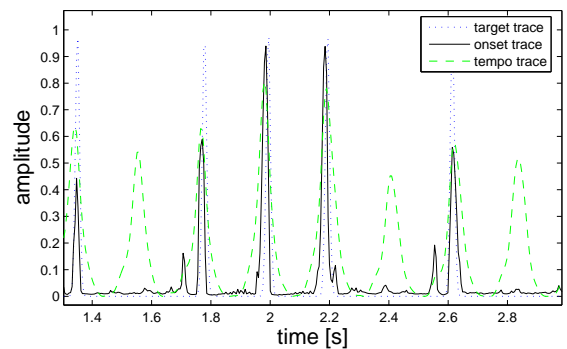


Figure 4: The target trace represents the ideal curve for the peak picking part. The onset trace represents the merged output of the neural network. The tempo trace, represent the global tempo information propagated to each local point.

2.2 Neural networks for the onset emphasizing part

As stated below, we use a neural network to combine the different transforms and to make frame classification. To train the network, we generate an onset target trace, from the labelled onset times. The onset trace is a mixture of Gaussians where the onset times are the means and the standard deviation is around 10 ms. See the green dashed curve of figure 3. The neural network is then convoluted to all times and frequencies and try to predict the onset trace.

2.2.1 Input variables

Onset patterns are translation invariant for the time axis i.e. the value of time doesn't help to tell whether it is an onset or not. However this is not totally true for frequency. It is translation invariant only for small frequency shift. Therefore, for big frequency shift, the neural network is unable to generalize. One solution is to provide the value of the frequency as input to the network but the chosen solution was to have a bigger window over the frequencies and therefore a smaller frequency shift. The red rectangle of figure 3 represent a typical input window for the neural network.

Now we have a window that makes 80% of the frequency spectrum and 100 ms width. This makes around 2000 variables for each plane. We need to make some dimensionality reduction. For each plane, we randomly took 80 variables inside the window, evenly distributed across the frequency and a Gaussian distribution across the time. This allows having a better resolution at the time we need to make the classification and some information in the future and in the past.

Now for each time and frequency of each song, we have two inputs of 80 variables. This makes a huge set of input for the learning algorithm which account for a big learning time, preventing us from making cross validation.

2.2.2 Neural network structure

We have used a feedforward neural network (FNN) with two hidden layers. The first hidden layer contained 18 neurons and the second 15 neurons. The learning algorithm was the conjugate gradient from the neural networks toolbox of Matlab. The dataset was separated into train and valid for hyperparameter estimation.

2.3 Peak picking

As we convolve the window across the frequency, we have more than one onset trace. The FNN is computed only for the frequency where the complete window can fit in the spec and we have adjusted the frequency step to have 10 windows for each time step. This makes 10 onset traces that are merged by averaging.

To find the peaks, we use a moving average of 500 ms width. This makes a mask to which is added some threshold. Every peak above mask + threshold is an onset. We adjust the threshold on the training set using a line search algorithm, to maximize the F-measure. A good line search algorithm is not necessary here. There is only one minimum, which is not far from the guess value and we don't need a good precision. We therefore only evaluated the function 10 times and took the best value.

2.4 Multi-nets algorithm

Now that we have the onset traces, we can find the rhythmic structure of the song, which can help resolve ambiguous peaks for the peak picking part. We also have trained different algorithms on different hyperparameters and merged everything with another neural network.

2.4.1 Merging information

We simply concatenate all the onset traces of all the networks and add a tempo trace for each network. We also add a trace that tells the arrhythmic state of the song. This trace is constant and is obtained from the entropy of the interonset histogram. This helps ignoring the tempo trace in case there is no tempo.

All this information merged together, makes another matrix with the same sampling rate as the spectrogram, but containing different information. Therefore the rest of the algorithm is exactly the same thing as the first one but the input come from the feature matrix and the input window takes 100% of the frequency spectrum.

For the many-nets, we've repeated the single net algorithm 5 times, varying the number of input variables, the window width, the frequency range, the frequency resolution... Those modifications should give each net different abilities, allowing them to be complementary. But we also believe that this setup lead to much overfitting. Unfortunately, the dataset we had was not sufficient to give a clear result about whether it is useful or not. Instead of doing this, we should implement adaboost for this classification task.

2.4.2 Tempo trace

We want to have a trace that brings the global information of the tempo back to each time frame. In other words, we would like that each value of the trace represent the probability that this particular time is in phase with the tempo.

One way of doing this is to take the interonset histogram of a particular point in the onset trace and compare it with the interonset histogram of all onsets. If both histograms are correlated, this means that this point is in phase with the tempo. But we have to find this for each point in the onset trace, which have a big computational cost.

In fact, the interonset histogram can be calculated simply by making the autocorrelation of the onset trace. In a similar way, we can calculate the tempo trace by taking the crosscorrelation of the onset trace with the interonset histogram. This is like making the third correlation of the onset trace with itself. Therefore, this tricorrelation operation, unlike the autocorrelation, gets the trace back in phase with the onset trace but this time, the tempo is propagated all over the place.

This gives a trace, where the peaks represent the probability that this particular time is part of a probable tempo and is also in phase with other peaks that are in the same tempo. This allows us to provide the same kind of information the brain have when it is expecting an onset. We finally let the final FNN to decide whether it is an onset or not according to the ambiguity of the peak and whether we are expecting a peak or not for this particular time. We also provide the entropy of the interonset histogram to give a hint whether the song have some rhythm or not.

In figure 4, we see that the onset trace have some small peaks that are not identified as being onsets. The green dashed curve represents the tempo trace and clearly the small peaks are unexpected onsets. Therefore, we can easily reject those peaks.

3 DATASET

For the dataset, we've used Pierre Leveau's dataset [Pierre Leveau and Richard (2004)] and we've added 5 polyphonic midi songs. The dataset was verified with a custom Matlab GUI, when the algorithm was giving errors.

The dataset was too small for the algorithm to have a good generalisation. Most of the hyperparameters values led to 95% of F-measure and most of the last 5% were onsets hidden behind signing or onsets generated by signing.

Therefore a better dataset would have helped for the

Algorithm	many-nets	single-net
Overall Average F-measure	80.07%	78.35%
Overall Average Precision	79.27%	77.69%
Overall Average Recall	83.70%	83.27%
Total Correct	7974	7884
Total False Positives	1776	2317
Total False Negatives	1525	1615
Total Merged	210	202
Total Doubled	53	60
Runtime (s)	4713	1022

Table 1: Overall results of the Mirex 2005 onset detection contest for participant Lacoste & Eck

learning of the algorithm.

4 RESULTS

The many-nets algorithm did slightly better than the single-net algorithm on the Mirex 2005 onset detection contest. The best one did 80.07% of F-measure and the second one did 78.35% of F-measure. See Table 1. The many-nets algorithm is complex, very slow and probably has a lot of overfitting. Therefore, the first one is probably more interesting, even if it is slightly less efficient.

The algorithm was design to perform on a wide range of music so it was less efficient than other algorithms on monophonic songs. But on the overall, many-nets and single-net algorithm were ranked respectively first and second.

The speed of the algorithm is incredibly slow, for tree reasons:

1. The many-nets algorithm contains 5 times the single-net algorithm. Therefore it is 5 times slower.
2. There was a bug in the M2k framework with Matlab itinerary on windows. For this reason, we had to remove the `-nosplash -nodesktop` options. Therefore, the complete Matlab desktop had to be loaded and closed for each songs.
3. Matlab codes are not optimal.

Then, implementing the single net algorithm in c would give a speed comparable to the other algorithms.

5 FUTURE WORKS

The algorithm was quit good at finding onsets times but there is still much room for improvements. We have 3 important improvements in head:

1. Testing the learning algorithm on a better dataset, like the Mirex 2005 onset detection dataset. This will allow better hyperparameters adjustment.
2. Trying adaboost as the classification algorithm.
3. Make some pre-processing on the spectrogram, to extract frame features, instead of working directly on the spectrogram.

6 CONCLUSION

Frame classification doesn't need to deal with too much information. Therefore not much dimensionality reduction is required, which account for very small pre-processing. For those reasons, neural networks are very suited for onset detection and they got the first rank on the Mirex 2005 onset detection contest, with 80.07% of F-measure.

If the algorithm is implemented in a suited way, it can be very fast and therefore, meet the goal of having fast and robust onset detection for a wide range of music. This would help most other algorithms that have to deal with music.

References

- J.P. Bello and M. Sandler. Phase-based note onset detection for music signals. In *proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-03)*, April 2003.
- Judith C. Brown. Calculation of a constant q spectral transform. *Journal of the Acoustical Society of America*, 89 (1):425–434, 1991.
- Laurent Daudet Pierre Leveau and Gael Richard. Methodology and tools for the evaluation of automatic onset detection algorithms in music. In *Proc. of the 5th International Conference on Music Information Retrieval (ISMIR)*, Barcelona, October 2004.