Review of LIF based spectrotemporal software.

Started 29 April 2019.

Last updated 1 May 2019.

File: spectrotemporal.m, Github d06ea7c..f5ca400

Overview: Sound signals (mono) from several files are subject to a spectrotemporal analysis that tries to find commonly occurring spectrotemporal patterns. The inputs are absolute values from Gammatone filtered bands, smoothed, onset and offset signals derived from these. The software is written so that other preprocessed datasets can be added as well. The inputs are used to create a windowed set of inputs (spatiotemporally, with a width of a number of time slices, and a height of the number of bands (N times K, programmatically)). These are fed to a set of leaky integrate-and-fire neurons, whose weights are updated when the activity in a neuron exceeds its threshold. The discovered patterns could be used as (higher-level) feature detectors, possibly choosing patterns discovered for different types of sounds.

Parameters:

SD: directory in which the files will be found

Filelistfile: list of file names for files to be processed.

Varargin parameters (with defaults):

Basic parameters for whole system filterbank

N = 100 ; % number of bandpass channels

M = 50 ; % number of LIF neurons ;

K = 30 ; % number of timesteps used

Gammatone Filterbank parameters

Fs = 44100 ; % sampling rate

minCochFreq = 200 ; % minimum gammatone frequency

maxCochFreq = 5000 ; %  maximum gammatone freuency

N\_erbs = 1 ; % default bandwidth

MAXDURATION = 100 ; % maximal duration of a single sound file

Smoothing, to avoid bumps from bandpassed signal

smoothlength = 0.01 ; % Bartlett filter parameter length of triangular (bartlett) window used to smooth rectified signal

Signals to use

useabs = true ; % will use the absolute value signal

useonset = false ; % will we use onset signals in processing?

useoffset = false ; % will we use offset signals in processing?

Parameters for onset and offset signal generation (if used)

sigma1 = 0.01 ; %half difference of Gaussians std

sigmaratio = 1.2 ; % ratio of the two HDOG's

nsamples = 4000 ; % number of samples used in onset/offset convolving function

Log transformation of signals

logabs = false ; % use log of absolute value

logonset = false ; % use log of onset/offset values

Leaky integrate and fire neuron parameters

LIFtimestep = 0.001 ; % timestep for use with LIF network

LIFdissipation = 100 ; % dissipation, 1/tau

LIFrp = 0.002 ; % refractory period

LIFthreshold = 1.0 ; % LIF neuron threshold

Weight initialisation

weightnorm = 1 ; % normalised value of weight

weightssupplied = false ; % default is to randomly initialise weights

Weight adaptation

k\_fired = 0.01 ; % adaptation learning rate for fired neuron

k\_notfired = 0.001 ; % adaptation learning rate for non-firing neurons

Processing, plus what might be changed.

1: file selection

A list of file names is created in the cell array filelist (read as [SD ‘/’ filelist{i}]).

2: bandpass filtering.

Gammatone filterbank is used, using function bmsigmono.m .

*If we were to use a parameter search, it would be good to only run the filterbank once, rather than once for each set of parameters.*

3: preprocessing.

Absolute value: each channel is full-wave rectified, the smoothed by convolution with a Bartlett filter of duration smoothlength. (this filter is precomputed).

Onset, offset signals: the absolute value signal is convolved with a half difference of Gaussians (again precomputed, parameters sigma1 and sigmaratio). The onset signal is the positive-going part of this, and the offset signal is the (inverted) negative-going part of this.

Compression: if logabs is set, the the log of (1 + absolute value) is used; if logonset is set, the same transform is applied to the onset and offset signals.

*Considering the text in Lyon’s book, it might be better to use a fractional power, rather than a log transform.*

4: Inputs to LIF neuron array.

Any combination of the absolute value signal, onset value signal and offset value signal can be used, depending on the logical values of useabs, useonset and useoffset. The values applied are re-packaged from the original sampling rate down to the LIFtimestep rate: this uses the MATLAB function resample.

*Note that the LIFtimestep defines the actual time period over which the signals are being interpreted: that is, it is K \* LIFtimestep.*

*Note that currently the inputs are always nonnegative. Should they be mean 0?*

5: weights for the LIF array.

The weights are a 3-dimensional array, with a 2D slice for each neuron. The overall weight array, weightarray is M (number of neurons) by N \* number of ranges (number of bandpass channels, repeated for each use of absolute value, onset and offset) by K (number of timesteps used).

Currently the weights are either supplied (using the weightssupplied varargin parameter) or are initialized randomly, by being set to values uniformly distributed between -1 and 1, then normalized so that the mean value for each neuron is 0, and the normalized value (root mean square) is weightnorm.

*Is this the best initialization? To begin with I had the weights all positive…there is an interaction between this and precisely how the inputs are supplied.*

6: LIF neurons.

The LIF neurons are characterized by the dissipation (LIFdissipation), threshold (LIFthreshold) and refractory period (LIFrp). The current value of the LIF neuron is LIFactivity, an M-vector, initialized to 0 at the start of each sound. They are all updated each LIFtimestep, by

1. (for each time instant) adding the summed value of the weight array time value times the input for that time instant over the input channels (B),
2. summing over the (K) time instants to produce an array (of length M) newLIFactivity
3. updating LIFactivity by adding (newLifactivity – (dissipation \* current LIFactivity) ) scaled by LIFtimestep.

LIF neurons fire if their activity exceeds the threshold (LIFthreshold, defaults to 1, alterable through varargin). Neurons that fire are unable to fire again until the end of their refractory period. They have their activity reset to 0, throughout the refractory period so that after this period their activity is reset to 0.

*Currently all the neurons share the same dissipation, threshold and refractory period. For neurons that do not fire, one could consider reducing the threshold. Note that the dissipation influences the time period over which the neurons integrate their input: this, the value of K and the value of LIFtimestep together determine the nature of the time-integration of the system.*

7: Adaptation.

At each time step (LIFtimestep) where any neuron fires, the weights are adapted. The intention is to make the neuron that fired more likely to fire again for similar input, and to make the neurons that did not fire less likely to fire for similar input (and more likely to fire for other inputs).

*This isn’t really what happens currently.*

Currently, for the neuron that fired: the weight array to that neuron is updated by adding k\_fired (constant, defaults to 0.01, settable using varargin) \* the signal that caused it to fire (i.e. the whole N by K array), followed by renormalization, that is, setting the weight array for that neuron to mean 0, and then making the norm of the weights to that neuron weightnorm.

For all the other neurons, the the weight array to the neuron is updated by subtracting k\_notfired (constant, defaults to 0.002, settable using varargin) \* the signal that caused the other neuron to fire (i.e. the whole N by K array), followed by renormalization, that is, setting the weight array for that neuron to mean 0, and then making the norm of the weights to that neuron weightnorm.

*All the neurons that fire at a particular timestep have their weights updated independently. So if multiple neurons fire, they are all independently updated.*

*Currently, all the inputs are nonnegative. Should they be mean 0? Or mean 0 at each timestep?*