A tutorial on Fourier-based Anechoic Demixing Algorithm (FADA)

Enrico Chiovetto[1]

1- Section for Computational Sensomotorics, Department of Cognitive Neurology, Hertie Institute for Clinical Brain Research, Centre for Integrative Neuroscience, University Clinic Tübingen, Tübingen, Germany.

enrico.chiovetto@uni.tuebingen.de
enrico.chiovetto@gmail.com

September 2016

Introduction

In this document we provide help and support for the use of the FADA toolbox for the unsupervised identification of motor primitives [1,2,3,4]. Theoretical derivation of the algorithm and its validation can be found in [5]. The toolbox is completely written in Matlab (Natick, MA) and can be utilised with Matlab version equal or higher than Matlab R2009b. The software is freeware. Below a complete list of the functions composing the toolbox is reported. For each function, a quick description is also given. In the last part of the document the reader can find a more detailed explanation on the use of the main functions of the toolbox to identify spatial, temporal, spatio-temporal (time-varying) and space-by-time primitives.

Complete list of functions:

Main functions

fada_spat.m: the function factorizes data according to a spatial organization.
fada_temp.m: the function factorizes data according to a temporal organization.
fada_tv.m: the function factorizes data according to a spatio-temporal organization.
fada_NM3F.m: the function factorizes data according to a space-by-time organization.

Additional functions

fadaSingleRun.m: this function runs only once the FADA algorithm as described in [], in the section *Implementation of the unifying algorithm*. Specifically, this function is valid for spatial, temporal and spatio-temporal factorizations.

fadaSingleRunNM3F.m : this function runs only once the FADA algorithm as described in [], in the section *Implementation of the unifying algorithm*. Such a function was written for data factorization into space-by-time organization.

weightsDelayEstimation.m: this function computes combination coefficients and delays for any type of primitives. When synchronous models are taken into consideration, a matrix of delays equal to zeros is returned.

FunModS.m: this function is minimized to update the absolute values of the Fourier coefficients of the primitives. Designed for spatial, temporal and spatio-temporal primitives.

FunModSNM3F.m: it has the same role of FunModS.m but for space-by-time factorizations.

FunPhaseS.m: this function is minimized to update the phases of the Fourier coefficients of the primitives. Designed for spatial, temporal and spatio-temporal primitives.
FunPhaseSNM3F.m: it has the same role of FunPhaseS.m but for space-by-time factorizations.

nonlinconModS.m: function implementing the non-negative constraints to fulfil when updating the absolute values of the Fourier coefficients of the primitives. It works for spatial, temporal and spatio-temporal primitives.

nonlinconModSNM3F.m: as nonlinconModS.m but for the identification of space-by-time primitives.

nonlinconPhaseS.m: function implementing the non-negative constraints to fulfil when updating the phases of the Fourier coefficients of the primitives. It works for spatial, temporal and spatio-temporal primitives.

nonlinconPhaseSNM3F.m: as nonlinconModS.m but for the identification of space-by-time primitives.

exp_vector.m: function computing a vector of exponentials.

syn_mix.m: this function mixes together spatial or temporal primitives after scaling them and, if needed, shifting them in time.

tv_syn_mix.m: function mixing together spatio-temporal (time-varying) primitives.

HarmonicFourierSeries.m: given a data matrix X (size ns-by-T, where ns is the number of signals and T is the number of time samples) is computes the first $N+1$ Fourier coefficients of each signal.

NNMF.m: this function performs non-negative matrix factorization on the input matrix X.

setNumA.m: this function finds the minimum number of harmonics (Fourier coefficients) needed to approximate the temporal signals given in the data matrix X (size ns-by-T, where ns is the number of signals and T is the number of time samples) given a minimum level of similarity between actual and reconstructed data at least equal to sim_lev (ranging between 0 and 1).

squeezeOne.m: remove singleton dimensions. It is a reviewed version of squeeze.m.

Use of the main functions

fada_spat.m

Identification of spatial primitives (model 1 in [5])

Syntax

```
[S, W, Rec, p] = fada_spat(X,T,NumS,NumA,Spos,Cpos)
[S, W, Rec, p] = fada_spat(X,T,NumS,NumA,Spos,Cpos,NumRuns)
[S, W, Rec, p] = fada_spat(X,T,NumS,NumA,Spos,Cpos,NumRuns,indep)
```

Definition

The function computes spatial primitives S and combination coefficients W. For this function the primitives are synchronous and cannot be time-shifted.

Description

```
Inputs: X    =  Matrix of the original data
                X is obtained by transposing the matrix with time signals
                of the same trial piled vertically and signals of different
                trials concatenated horizontally. X will thus have size
                (NumT x T) x (NumM). NumT is the number of trials, NumM
                the number of degrees of freedom recorded in each trial,
                for instance the number of muscles.
                In this case the signals of each trial do not need to be
                standardized in time.
        NumM  = Number of degrees of freedom
        NumS  = Number of primitives to learn
        NumA  = Number of Fourier harmonics to approximate the original
                signals
        Spos  = positivity of the synergies 0 = unconstrained
                                            1 = positive
        Cpos  = positivity of the combination coefficients
                                            0 = unconstrained
                                            1 = positive
        NumRuns = number of runs
        indep = independence of the sources  'ica' = independent sources
                                             'unc' = uncorrelated sources
                                             'lsq' = Least-square approx..


  Outputs:
         S    = identified primitives
         W    = identified combination coefficients
        Rec   = reconstructed data
         p    = variance accounted for
```

By default `NumRuns` is set to 5 and `indep` to `'lsq'`. The minimum and maximum number of iterations of can be set in fadaSingleRun.m by modifying `miniter` and `maxiter`. Default values are respectively 10 and 50.

Example 1:

This example shows how to use fada_spat to extract unconstrained spatial primitives and weights from the data matrix Mix. Such a matrix is obtained by linear combination of four spatial, synchronous primitives (matrix Primitives) scaled by a mixing matrix Weights.

```
load Sim_unconstrained_synchronous_mixture.mat;% load the data
T = 10;                                % number of degrees of freedom
NumS = 4;                              % number of primitives
NumA = 5;                              % number of harmonics
Spos = 0;                              % prior on the primitives
Cpos = 0;                              % prior on the weights
NumRuns=5;                             % Number of runs
% extraction of spatial primitives with unconstrained values. 5 harmonics, 5
runs, least-square approximation.
[S, W, Rec, p] = fada_spat(Mix,T,NumS,NumA,Spos,Cpos,NumRuns,'lsq');
```

fada_temp.m (models 2 and 3 in [5])
Identification of temporal primitives

Syntax

```
[S, W, Tau, Rec, p] = fada_temp(X,T,NumS,NumA,Spos,Cpos,del)
[S, W, Tau, Rec, p] = fada_temp(X,T,NumS,NumA,Spos,Cpos,del,NumRuns)
[S, W, Tau, Rec, p] = fada_temp(X,T,NumS,NumA,Spos,Cpos,del,NumRuns,indep)
```

Definition

The function computes temporal primitives S, combination coefficients W and temporal delays Tau.

Description

```
Inputs: X    = Matrix of the original data.
               When one-dimensional temporal motor primitives are to
               be idenfied X has to be a matrix with time signals piled
               vertically. X will thus have size (NumT x NumM) x T. NumT
               is the number of trials, NumM the number of degrees of
               freedom recorded in each trial, for instance the number
               of muscles. All signals need to be standardized in time
               in a pre-processing phase.
        T     = Number of time samples per signal
        NumS  = Number of primitives to learn
        NumA  = Number of Fourier harmonics to approximate the original
                signals
        Spos  = positivity of the synergies 0 = unconstrained
                                            1 = positive
        Cpos  = positivity of the combination coefficients
                                            0 = unconstrained
                                            1 = positive
        del   = modelling of the delays     0 = non-shiftable primitives
                                            1 = time shiftable primitives
        NumRuns = number of runs
        indep = independence of the sources  'ica' = independent sources
                                             'unc' = uncorrelated sources
                                             'lsq' = Least-square approx.
  Outputs:
         S    = identified primitives
         W    = identified combination coefficients
        Tau   = identified delays of the primitives
        Rec   = reconstructed data
         p    = variance accounted for
```

By default `NumRuns` is set to 5 and `indep` to `'lsq'`. The minimum and maximum number of iteration of can be set in fadaSingleRun.m by modifying `miniter` and `maxiter`. Default values are respectively 10 and 50.

Example 1:

This example shows how to use fada_temp to factorize a non-negative data matrix Mix into an anechoic organization []. Such a matrix is obtained by linear combination of four temporal primitives (Primitives) scaled by a mixing matrix Weights and shifted in time according to the delays in the matrix Delays.

```
load Sim_non-negative_anechoic_mixture.mat    % load the data
T = 100;                                      % number of time samples
NumS = 4;                                     % number of primitives
NumA = 25;                                    % number of harmonics
Spos = 1;                                     % non-negativity prior on the primitives
Cpos = 1;                                     % non-negativity prior on the weights
Del = 1;                                      % no delay
NumRuns=5;                                    % number of runs
% extraction of temporal non-negative primitives shifted in time. 5
harmonics, 5 runs, least-square approximation.
[S, W, Tau, Rec, p] = fada_temp(Mix,T,NumS,NumA,Spos,Cpos,Del,NumRuns,'lsq');
```

fada_tv.m (model 4 in [5])
identification of spatio-temporal primitives.

Syntax

```
[S, W, Tau, Rec, p] = fada_tv(X,T,NumS,NumA,Spos,Cpos,del)
[S, W, Tau, Rec, p] = fada_tv(X,T,NumS,NumA,Spos,Cpos,del,NumRuns)
[S, W, Tau, Rec, p] = fada_tv(X,T,NumS,NumA,Spos,Cpos,del,NumRuns,indep)
```

Definition

The function performs Fourier-based demixing of the input data and extracts time-varying, spatio-temporal primitives and corresponding weights and delays.

Description

```
Inputs: X    =  Matrix of the original data.
                X has to be a matrix with time signals of the
                same trial piled vertically and signals of different
                trials concatenated horizontally. X will thus have size
                (NumM) x (NumT x T).  NumT is the number of trials, NumM
                the number of degrees of freedom recorded in each trial,
                for instance the number of muscles). All signals need to
                be standardized in time in a pre-processing phase.
        T     = Number of time samples per signal
        NumS  = Number of primitives to learn
        NumA  = Number of Fourier armonics to approximate the original
        signals
        Spos  = positivity of the synergies 0 = unconstrained
                                            1 = positive
        Cpos  = positivity of the combination coefficients
                                            0 = unconstrained
                                            1 = positive
        del   = modelling of the delays     0 = non-shiftable synergies
                                            1 = time shiftable synergies
        NumRuns = number of runs
        indep = independence of the sources  'ica' = independent sources
                                             'unc' = uncorrelated sources
                                             'lsq' = Least-square approx.

  Outputs:
         S    = identified synergies
         W    = identified combination coefficients
        Tau   = identified delays
        Rec   = reconstructed data
         p    = variance accounted for
```

By default `NumRuns` is set to 5 and `indep` to `'lsq'`. The minimum and maximum number of iteration of can be set in fadaSingleRun.m by modifying `miniter` and `maxiter`. Default values are respectively 10 and 50.

Example 1:

This example shows how to use fada_tv to factorize non-negative data matrix Mix according to a spatio-temporal organization. Such a matrix is obtained by linear combination of four spatio-temporal primitives (Primitives) scaled by a mixing matrix Weights and time-shifted according to the delays in the matrix Delays.

```
load Sim_non-negative_spatio_temporal_mixture.mat    % load the data
T = 100;                                % number of time samples
NumS = 4;                               % number of primitives
NumA = 25;                              % number of harmonics
Spos = 1;                               % non-negativity prior on the primitives
Cpos = 1;                               % non-negativity prior on the weights
Del = 1;                                % no delay
% extraction of spatio-temporal, non-negative primitives shifted in time. 5
harmonics, 5 runs, least-square approximation (default).
[S, W, Tau, Rec, p] = fada_tv(Mix,T,NumS,NumA,Spos,Cpos,Del);
```

fada_NM3F.m (model 5 in [5])

Identification of spatial and temporal primitives, according to a space-by-time organization.

Syntax

```
[St, Ss, W, Tau, Rec, p] = fada_NM3F(X,T,NumS,NumW,NumA,del)
[St, Ss, W, Tau, Rec, p] = fada_NM3F(X,T,NumS,NumW,NumA,del,NumRuns)
```

Definition

The function computes temporal primitives S, spatial primitives Ss, combination coefficients W and time delays Tau. All the parameters provided by the function are non-negative.

Description

```
Inputs: X    =  Matrix of the original data.
                X has to be a matrix with time signals of the
                same trial piled vertically and signals of different
                trials concatenated horizontally. X will thus have size
                (NumM) x (NumT x T).  NumT is the number of trials, NumM
                the number of degrees of freedom recorded in each trial,
                for instance the number of muscles.
                All signals need to be standardized in time in a
                pre-processing phase.
        T     = Number of time samples per signal
        NumS  = Number of temporal primitives
        NumW  = Number of spatial primitives
        NumA  = Number of Fourier harmonics to approximate the original
        signals
        del   = modelling of the delays     0 = non-shiftable primitives
                                            1 = time shiftable primitives
        NumRuns = number of runs

  Outputs:
        St    = identified temporal primitives
        Ss    = identified spatial primitives (size NumM x NumW)
         W    = identified combination coefficients
        Tau   = identified delays
        Rec   = reconstructed data
         p    = variance accounted for
```

By default `NumRuns` is set to 5 and `indep` to `'lsq'`. The minimum and maximum number of iteration of can be set in fadaSingleRun.m by modifying `miniter` and `maxiter`. Default values are respectively 10 and 50.

Example 1:

This example shows how to use fada_NM3F to factorize a non-negative data matrix NMF_mix. Such a matrix is obtained by linear combination of four temporal primitives (Primitives) scaled by a mixing matrix Weights.

```
load Sim_space_by_time_mixture.mat    % load the data
T = 100;                              % number of time samples
NumS = 2;                             % number of temporal primitives
NumW = 3;                             % number of spatial primitives
NumA = 25;                            % number of harmonics
Del = 1;                              % no delay
NumRuns = 5;                          % number of runs
% factorization of data matrix Mix into a space-by-time organization. 5 runs,
% least-square approximation (default)
[S, Ss, W, Tau, Rec, p] = fada_NM3F(Mix,T,NumS,NumW,NumA,Del,NumRuns);
```

Additional functions

weightsDelayEstimation.m

Identification of weights and delays.

Syntax

```
[outW,outD,outRec] = WeightsDelayEstimation(data,prim,W,D,delay_box,Cpos,del)
```

Definition

weightsDelayEstimation computes the weights and delays given the primitives. Weights and delays are optimized so as to minimize the difference between original and reconstructed data. Within the function, the variable `maxiter` indicates the number of times the loop to update the weights and delays is executed (step 2 of page 10 in [5]). By default `maxiter = 250`.

Description

```
Inputs:
       data      =  Matrix of the original data. It has size NumM x T,where
                    NumM is the number of degrees of freedom recorded in each
                    trial, for instance the number of muscles. T the number
                    of time samples).

       prim      =  Given primitives
       W         =  Initial values of the weights
       D         =  Initial values of the delays
       delay_box =  column array containing the upper bound of the delays.
                    lower bounds are all set to zero. The vector has NumS
                    elements, where NumS is the number of primitives.
       Cpos      =  positivity of the combination coefficients
                                          0 = unconstrained
                                          1 = positive
       del       =  modelling of the delays  0 = non-shiftable primitives
                                          1 = time shiftable primitives

Outputs:
       outW      =  updated combination coefficients
       outD      =  updated delays
       outRec    =  reconstructed data
```

Terms of use

If you use this software for your research, please refer to [5].

References

1. Bizzi E., Cheung V. C. K., d'Avella A., Saltiel P., Tresch M. (2008). Combining modules for movement. Brain Res. Rev. 57, 125–133 10.1016/j.brainresrev.2007.08.004 .
2. Flash, T., and Hochner, B. (2005). Motor primitives in vertebrates and invertebrates. Curr. Opin. Neurobiol. 15, 660–666. doi: 10.1016/j.conb.2005.10.011.
3. Tresch M.C., Jarc A. (2009). The case for and against muscle synergies. Curr Opin Neurobiol 19:601–607.
4. Chiovetto E., Berret B., Delis I., Panzeri S., Pozzo T. (2013). Investigating reduction of dimensionality during single-joint elbow movements: a case study on muscle synergies. Front. Comput. Neurosci. 7:11 10.3389/fncom.2013.00011.
5. Chiovetto E, d'Avella A and Giese MA. (2016). A Unifying Framework for the Identification of Motor Primitives. arXiv preprint arXiv:1603.06879.