User Manual Hierarchical Unsupervised Generative Embedding

Yu Yao

Translational Neuromodeling Unit
Institute for Biomedical Engineering
University of Zurich & ETH Zurich

August 12, 2020

Contents

1 Introduction

This document contains the user guide for the HUGE toolbox, which is part of TAPAS (Translational Algorithms for Psychiatry-Advancing Science), a collection of tools developed at the Translational Neuromodeling Unit, Zurich. This guide provides a detailed documentation of the user interface of the HUGE toolbox. To get started quickly, you may run the examples in the demo script `tapas_huge_demo.mlx`.

HUGE stands for Hierarchical Unsupervised Generative Embedding. It is a generative model for (task-based) fMRI data. It can be used to stratify hetero-

geneous cohorts into subgroups or learn the prior distribution of DCM parameters from data via empirical Bayes. For more details on the theory behind the HUGE model, please refer to Yao et al. [2018].

To use this toolbox, change your current working directory to the directory that contains the `@tapas_Huge` folder or add this directory to your Matlab path.

2 The `tapas_Huge` Class

This toolbox implements the HUGE model as a Matlab class called `tapas_Huge`. Data, options and results are stored in class properties, while the main functionalities of the toolbox are provided by class methods. Additional functionalities are provided by regular Matlab functions.

Instances of the `tapas_Huge` class are created by calling the class constructor `tapas_Huge()`. The constructor is a function that can be called without arguments, which creates an empty `tapas_Huge` object:

```
obj = tapas_Huge()
```

However, the constructor also accepts optional arguments in the form of name-value pairs. For example, one can add a short description using the `tag` property

```
obj = tapas_Huge('Tag','my model')
```

or import data using the `Dcm` property

```
obj = tapas_Huge('Dcm',dcms)
```

For more examples, see the demo script `tapas_huge_demo.mlx`.

3 Class Properties

Each instance of the `tapas_Huge` class stores data, options and results in class properties, which are documented below. Properties can be accessed using dot notation:

```
value = obj.property
```

where `obj` is a class instance and `property` is the name of the property.

Note that all properties, except for `K` (the number of clusters) and `tag` are read only, i.e. they cannot be changed by the user directly. Instead, their value is set automatically when importing data using the `import` method or inverting the model using the `estimate` method.

Below is a list of all properties of the `tapas_Huge` class.

K

A scalar integer containing the number of clusters of the HUGE model.

tag

A character array that can be used for storing a short description of the model.

L

A scalar integer containing the number of experimental inputs (i.e., the dimensionality of `obj.inputs.u`).

M

A scalar integer containing the number of group-level confound variables (like age, sex, handedness, etc).

N

A scalar integer containing the number of subjects. Permissions: read only.

R

A scalar integer containing the number of brain regions in the model.

idx

A Matlab struct storing parameter indices with the following fields:

`clustering`	Vector of indices of DCM parameters used for clustering.
`homogenous`	Vector of indices of homogenous parameters. I.e. DCM parameters which are estimated but not used for clustering.
`P_c`	Number of clustering parameters.
`P_c`	Number of homogeneous parameters.
`P_f`	Number of parameters in a fully connected DCM with the same number of regions as the current one.

dcm

A Matlab struct in SPM's DCM format containing the DCM network structure.

inputs

A struct array containing the experimental stimuli for all subject, with two fields:

u	Array containing experimental stimuli.
dt	Sampling time interval.

data

A struct array containing the fMRI data for all subjects, with fields:

bold	Array containing the BOLD time series.
te	The echo time (TE).
tr	The repetition time TR.
X0	Vector containing subject-level confounds.
res	The residual forming matrix of X0.
confounds	Vector containing the group-level confounds (e.g., sex, age, etc).
spm	A struct containing the posterior from SPM (The Ep field in SPM's DCM format).

labels

A struct containing names for regions and inputs.

options

A struct storing options for the HUGE model. Note: Set options using name-value pair arguments with the `taps_huge` and `estimate` methods.

prior

A struct storing priors of the HUGE model. Note: Set priors using name-value pair arguments with the `taps_huge` and `estimate` methods.

posterior

A struct storing the estimation results. The fields correspond to the parameters of the posterior distribution (seeYao et al. [2018]).

`alpha`	Vector of posterior cluster weights (α).
`m`	Array of posterior cluster means (m).
`tau`	Vector containing the τ parameter of the inverse-Wishart posterior distribution.
`nu`	Vector of posterior degree of freedom parameters (ν).
`S`	Array containing the scale matrices of the inverse-Wishart posterior distribution (S).
`q_nk`	Array of posterior assignment probabilities (q_{nk}).
`mu_n`	Array of posterior mean subject-level DCM parameters (μ_n).
`Sigma_n`	Array of posterior covariances of subject-level DCM parameters (Σ_n).
`b`	Vector containing the b parameter of the posterior Gamma distribution over noise precision.
`a`	Vector containing the a parameter of the posterior Gamma distribution over noise precision.
`m_beta`	When using group-level confounds, array containing posterior mean of coefficients, (m_β). Otherwise, empty array.
`S_beta`	When using group-level confounds, array containing posterior covariance of coefficients (S_β). Otherwise, empty array.
`nfe`	The negative free energy (F).
`mu_r`	When using group-level confounds, array containing posterior mean of residual DCM parameters, (m_r). Otherwise, empty array.
`Sigma_r`	When using group-level confounds, array containing posterior covariance of residual DCM parameters, (S_r). Otherwise, empty array.
`nrv`	Normalized residual variance. An array containing the amount of non-explained variance (1 minus variance explained) per subject and per region.
`bPurity`	When using synthetic data, balanced purity of estimation result.
`method`	Character array indicating inversion method.
`version`	Character array indicating toolbox version.
`seed`	Random number generator seed.

trace

A struct containing convergence diagnostics, with fields:

`nfe`	Vector containing the history of the negative free energy during iterative VB inversion.

`nDcmUpdate`	Vector containing the number of times the subject-level DCM parameter update was accepted.
`convergence`	The number of iterations it took VB to converge.
`kmeans`	Struct containing information about the K-means initialization step.
`epsilon`	Cell array containing the residual (observed minus predicted BOLD time series) for each subject at convergence.

`aux`

A struct used for storing auxiliary variables during model estimation.

`model`

When using the model to simulate data, this struct contains the model parameters used for generating the synthetic dataset. Fields correspond to HUGE model parameters:

`pi`	Vector of cluster weights (π).
`d`	Array of cluster labels in one-hot encoding (d).
`mu_k`	Array of cluster means (μ_k).
`Sigma_k`	Array of cluster covariances (Σ_k).
`mu_h`	Vector containing mean of homogenous DCM parameters (μ_h).
`Sigma_h`	Array containing covariance of homogenous DCM parameters (Σ_h).
`theta_c`	Array of clustering DCM parameters (θ_c).
`theta_h`	Array of homogenous DCM parameters (θ_h).
`lambda`	Array of measurement noise precisions (λ).
`x_n`	Array of group-level confounds (e.g. age, sex, etc. x_n).
`beta`	Array of confound coefficients (β).
`options`	Struct containing options for simulation.
`seed`	Random number generator seed.

`constants`

Struct storing constants used by the model.

`version`

Character array indicating toolbox version.

4 Class Methods

The main functionalities of the HUGE toolbox are implemented as methods of the `tapas_Huge` class. These methods are documented below. There are two equivalent ways to call class methods:

```
obj = obj.method( ... )
```

or

```
obj = method( obj, ... )
```

where `obj` is an instance of the `tapas_Huge` class and `method` is the class method you want to call.

tapas_Huge.estimate

```
Estimate parameters of the HUGE model.

INPUTS:
  obj - A tapas_Huge object containing fMRI time series.

OPTIONAL INPUTS:
  This function accepts optional name-value pair arguments. For a list of
  valid name-value pairs, see the user manual or type 'help
  tapas_huge_property_names'.

OUTPUTS:
  obj - A tapas_Huge object containing the estimation result in the
        'posterior' property.

EXAMPLES:
  [obj] = ESTIMATE(obj)    Invert the HUGE model stored in obj.

  [obj] = ESTIMATE(obj, 'K', 2)    Set the number of clusters to 2 and
      invert the HUGE model stored in obj.

  [obj] = ESTIMATE(obj, 'Verbose', true)    Print progress of estimation
      to command line.

  [obj] = ESTIMATE(obj, 'Dcm', dcms, 'OmitFromClustering', 1)    Import
      data stored in 'dcms' and omit self-connections from clustering.

See also TAPAS_HUGE_DEMO
```

Note that the HUGE toolbox uses a parameterization such that the DCM networks are self-inhibiting by default. This is achieved by subtracting 0.5 from the self-connections. Hence, specifying a prior mean of zero for all DCM connections implies that the effective self-connectivity is -0.5. This is similar to DCM self-connections in SPM12. This parametrization has been chosen for the convenience of the user, since it eliminates the need to identify the position of the self-connections in the parameter vector.

tapas_Huge.simulate

```
Generate synthetic task-based fMRI time series data, using HUGE as a
generative model.

INPUTS:
  obj      - A tapas_Huge object.
  clusters - A cell array containing DCM structs in SPM's DCM format,
             indicating the DCM network structure and cluster mean
             parameters.
  sizes    - A vector containing the number of subjects for each cluster.

OPTIONAL INPUTS:
  This function accepts optional name-value pair arguments. For a list of
  valid name-value pairs, see examples below.

OUTPUTS:
  obj - A tapas_Huge object containing the simulated fMRI time series in
        its 'data' property and the ground truth parameters in its
        'model' property.

EXAMPLES:
  [obj] = SIMULATE(obj,clusters,sizes)    Simulate fMRI time series with
      cluster mean parameters given in 'clusters' and number of subjects
      given in 'sizes'.

  [obj] = SIMULATE(obj,clusters,sizes,'Snr',1)    Set signal-to-noise
      ratio of fMRI data to 1.

  [obj] = SIMULATE(obj,clusters,sizes,'NoiseFloor',0.1)    Set minimum
      noise variance to 0.1.

  [obj] = SIMULATE(obj,clusters,sizes,'confounds',confounds)    Introduce
      group-level confounds (like sex or age).

  [obj] = SIMULATE(obj,clusters,sizes,'beta',beta)    Set coefficients
      for group-level confounds.
```

```
[obj] = SIMULATE(obj,clusters,sizes,'variant',2)    Set confound
    variant to 2 (i.e., clusters do not share confound coefficients).

[obj] = SIMULATE(obj,clusters,sizes,'Inputs',U)    Introduce subject-
    specific experimental stimuli. 'U' must be an array or cell array
    of structs with fields 'dt and 'u'.

[obj] = SIMULATE(obj,clusters,sizes,'OmitFromClustering',omit)
    Designate DCM parameters to be excluded from clustering model.
    Excluded parameters still exist in the DCM network structure, but
    are drawn from the same distribution for all subjects. 'omit'
    should be a struct with fields a, b, c, and/or d which are bool
    arrays with sizes matching the corresponding fields of the DCMs. If
    omit is an array, it is interpreted as the field a. If omit is 1,
    it is expanded to an identity matrix of suitable size.
```

tapas_Huge.plot

```
Plot cluster and subject-level estimation result from HUGE model.

INPUTS:
  obj - A tapas_Huge object containing estimation results.

OPTIONAL INPUTS:
  subjects - A vector containing indices of subjects for which to plot
             detailed results.

OUTPUTS:
  fHdl - Handle of first figure.

See also tapas_Huge.ESTIMATE
```

tapas_Huge.save

```
Save properties of HUGE object to mat file.

INPUTS:
  filename - File name.
  obj      - A tapas_Huge object.
```

```
OPTIONAL INPUTS:
  Names of property to be saved. See examples below.

EXAMPLES:
  SAVE(filename,obj)    Save properties of 'obj' as individual variables
      file specified in 'filename'.

  SAVE(filename,obj,'options','posterior','prior')    Only save the
      'options', 'posterior' and 'prior' properties of 'obj'.
```

tapas_Huge.import

```
Import fMRI time series data into HUGE object.

  WARNING: Importing data into a HUGE object will delete any data and
  results which are already stored in that object.

INPUTS:
  obj  - A tapas_Huge object.
  dcms - A cell array containing DCM structs in SPM's DCM format.

OPTIONAL INPUTS:
  confounds - Group-level confounds (e.g., age, sex, etc). 'confounds'
              must be empty or an array with as many rows as there are
              elements in 'dcm'.
  omit      - specifies DCM parameters which should be omitted from
              clustering. Parameters omitted from clustering will still
              be estimated, but under a static Gaussian prior. 'omit'
              should be a struct with fields a, b, c, and/or d which are
              bool arrays with sizes matching the corresponding fields of
              the DCMs. If omit is an array, it is interpreted as the
              field a. If omit is 1, it is expanded to an identity matrix
              of suitable size.
  append    - bool, if true keep current fMRI time series and append new
              data in 'dcms'. Note: estimation results will still be
              deleted.

OUTPUTS:
  obj - A tapas_Huge object containing the imported data.

EXAMPLES:
  [obj] = IMPORT(obj,dcms)    Import the fMRI time series and DCM
      network structure stored in dcms into obj.
```

```
  [obj] = IMPORT(obj,dcms,confounds)    Import group-level confounds
      (like age or sex) in addition to fMRI data.

  [obj] = IMPORT(obj,dcms,[],1)    Import fMRI data and network
      structure. Exclude self-connections from clustering.

See also tapas_Huge.REMOVE, tapas_Huge.EXPORT
```

tapas_Huge.export

```
Export results and data from HUGE object to SPM's DCM format.

INPUTS:
  obj - A tapas_Huge object containing data.

OUTPUTS:
  dcms      - A cell array containing DCM structs in SPM's DCM format.
  confounds - An array containing group-level confounds (like age or
              sex) if available. 'confounds' is an array with one row
              per subject.

EXAMPLES:
  [dcms] = EXPORT(obj)    Export fMRI time series and estimation results
      stored in obj.

  [dcms,confounds] = EXPORT(obj)    Also export group-level confounds
      (like age or sex).

See also tapas_Huge.IMPORT
```

tapas_Huge.remove

```
Remove data (fMRI time series, confounds, DCM network structure, ... )
and estimation results from HUGE object.

INPUTS:
  obj - A tapas_Huge object.

OPTIONAL INPUTS:
```

```
  idx - Only remove data of the subjects indicated in 'idx'. 'idx' must
        be a vector containing numeric or logical array indices.

OUTPUTS:
  obj - A tapas_Huge object with data and results removed.

EXAMPLES:
  [obj] = REMOVE(obj)    Remove results and data of all subjects.

  [obj] = REMOVE(obj,1:5)    Remove results and data for first 5
                             subjects.

  [obj] = REMOVE(obj,'all') is the same as [obj] = REMOVE(obj)

  [obj] = REMOVE(obj,0) is the same as [obj] = REMOVE(obj)

See also tapas_Huge.IMPORT
```

5 Name-Value Pair Arguments

Both the class constructor `tapas_Huge` and the method `estimate` accept optional arguments in the form of name-value pairs. Options set using name-value pair arguments are persistent across the lifetime of the object. Below is a list of valid name-value pairs that can be used with these two functions.

Name:	`Confounds`
Value:	double array
Description:	Specify confounds for group-level analysis (e.g. age or sex) as double array with one row per subject and one column per confound. Note: This property can only be used in combination with the `Dcm` property. WARNING: This feature is experimental and has not been extensively tested.

Name:	`ConfoundsVariant`
Value:	`'none'` | `'global'` | `'cluster'` (default: `'global'` if confounds specified, `'none'` otherwise)
Description:	Determines how confounds enter model. `'none'`: Confounds are not used. `'global'`: Confounds enter global regression (variant 1). `'cluster'`: Confounds enter cluster-specific regression (variant 2).

Name:	`Dcm`
Value:	cell array of DCM structs in SPM format
Description:	Specify DCM structure and BOLD time series for all subjects as cell array with one DCM struct in SPM format per subject.

Name:	`K`
Value:	positive integer (default: 1)
Description:	Number of clusters.

Name:	`Method`
Value:	`'VB'`
Description:	Name of inversion method specified as character array. VB: variational Bayes

Name:	`NumberOfIterations`
Value:	positive integer (default: 999)
Description:	Maximum number of iterations of VB scheme.

Name:	`OmitFromClustering`
Value:	array of logical | struct with fields `a`, `b`, `c` and `d` (default: empty struct)
Description:	Select DCM parameters to exclude from clustering. Parameters excluded from clustering will still be estimated, but under a static Gaussian prior. If input is array, it will be treated as the `a` field of a struct. Missing fields will be treated the same as arrays of `false`. Note: This property can only be used in combination with the `Dcm` property.

Example:	Exclude the self-connections from clustering:
	`obj = obj.estimate('OmitFromClustering', 1);`
	Exclude input strength from clustering:
	`omit = struct('c',obj.dcm.c));`
	`obj = obj.estimate('OmitFromClustering', omit);`

Name:	`PriorClusterMean`
Value:	`'default'` | row vector of double
Description:	Prior cluster mean. Scalar input will be expanded into vector. Default: `[0, ... ,0]\verb`.

Name:	`PriorClusterVariance`
Value:	`'default'` | symmetric, positive definite matrix of double
Description:	Prior mean of cluster covariances. Must be a symmetric, positive definite matrix. Scalar input will be expanded into diagonal matrix. Default: `diag([0.01, ..., 0.01])`.

Name:	`PriorDegree`
Value:	`'default'` | positive double
Description:	ν_0 determines the prior precision of the cluster covariance. For VB, this is the degrees of freedom of the inverse-Wishart. Default: `100`.

Name:	`PriorVarianceRatio`
Value:	`'default'` | positive double
Description:	Ratio τ_0 between prior mean cluster covariance and prior covariance over cluster mean. Prior covariance over cluster mean equals prior cluster covariance divided τ_0. Default: `0.1`.

Name:	`Randomize`
Value:	bool (default: `false`)
Description:	If `true`, starting values for subject level DCM parameter estimates are randomized.

Name:	`SaveTo`
Value:	character array
Description:	Location for saving results consisting of path name and optional file name. Path name must end on file separator and point to an existing directory. If file name is not specified, it is set to 'huge' followed by date and time.

Name:	`Seed`
Value:	double | cell array of double and rng name | random number generator seed obtained with `rng` command
Description:	Seed for random number generator.

Name:	`StartingValueDcm`
Value:	`'default'` | `'spm'` | double array (default: `'default'`)
Description:	Starting values for subject-level DCM parameter estimates. `'default'` uses prior cluster mean for all subjects. 'spm' uses values supplied in the 'Ep' field of the SPM DCM structs. Use a double array with number of rows equal to number of subjects to specify custom starting values.

Name:	`StartingValueGmm`
Value:	`'default'` | double array (default: `'default'`)
Description:	Starting values for cluster-level DCM parameter estimates. `'default'` uses prior cluster mean for all clusters. Use a double array with number of rows equal to number of clusters to specify custom starting values.

Name:	`Tag`
Value:	character array
Description:	Model description

Name:	`TransformInput`
Value:	bool (default: false)
Description:	Transform input strength (C matrix) to log-domain.

Name:	`Verbose`
Value:	bool (default: `false`)
Description:	Activate/deactivate command line output.

6 Other Functions

The HUGE toolbox provides additional functionalities in the form of regular Matlab functions, which are documented below. These function are used internally by the `tapas_Huge` class. However, some of them may be useful for the user.

`tapas_huge_boxcar`

```
Generate a boxcar function for use as experimental stimulus. All
timing-related arguments must be specified in seconds.

INPUTS:
  dt      - Numeric scalar indicating sampling time interval.
  nBoxes  - Vector indicating number of blocks.
  period  - Vector containing time interval between block onsets.
```

```
  onRatio - Vector containing ratio between block length and 'period'.
            Must be between 0 and 1.

OPTIONAL INPUTS:
  padding - Length of padding at the beginning and end.

OUTPUTS:
  u - A cell array containing the boxcar functions.

EXAMPLES:
  u = TAPAS_HUGE_BOXCAR(.01, 10, 3, 2/3, [4 0])    Generate boxcar
      function with 10 blocks, each 2 seconds long with 1 second inter
      block interval and onset of first block at 4 seconds.

See also tapas_Huge.SIMULATE
```

tapas_huge_bold

```
Integrates the DCM forward equations to generate the predicted fMRI bold
time series.

INPUTS:
  A, B, C, D - DCM connectivity matrices.
  tau        - Venous transit time.
  kappa      - Decay of vasodilatory signal.
  epsilon    - Ratio of intra- and extravascular signal.
  R          - Number of regions.
  u          - Experimental stimuli.
  L          - Number of experimental stimuli.
  E_0        - Resting oxygen extraction fraction.
  r_0        - Slope of intravascular relaxation rate.
  V_0        - Resting venous volume.
  vartheta_0 - Frequency offset at the outer surface of magnetized
               vessels (Hz).
  alpha      - Grubb's exponent.
  gamma      - rate constant of feedback regulation.
  TR         - Repetition time.
  TE         - Echo time.
  dt         - Sampling interval of inputs.

OUTPUTS:
  response - matrix of predicted response for each region
                 (column-wise)
```

```
  x        - time series of neuronal states
  s        - time series of vasodilatory signal
  f1       - time series of flow
  v1       - time series of blood volume
  q1       - time series of deoxyhemoglobin content.
```

tapas_huge_bpurity

```
Calculate balanced purity (see Brodersen2014 Eq. 13 and 14) for a set of
ground truth labels and a set of estimated labels

INPUTS:
  labels    - Vector of ground truth class labels.
  estimates - Clustering result as array of assignment probabilities or
              vector of cluster indices.

OUTPUTS:
  balancedPurity - Balanced purity score according to Brodersen (2014)

EXAMPLES:
  bp = TAPAS_HUGE_BPURITY(labels,estimates)
```

For more information on the fMRI BOLD model, please refer to Stephan et al. [2007].

tapas_huge_compile

This function is used internally to compile the mex function the HUGE toolbox needs. This happens automatically the first time you use the HUGE toolbox. In general, there is no need to call this function manually. However, you need to make sure that a C compiler is installed on your system. More details on this topic, including links to freely available compilers, can be found at:

`https://www.mathworks.com/support/requirements/supported-compilers.html`

tapas_huge_logdet

This function is intended for internal use only. Do not call directly

```
Numerical stable calculation of log-determinant for positive-definite
matrix.

INPUT:
  A - Positive definite matrix.

OUTPUT:
  ld - log(det(A))

EXAMPLE:
  ld = TAPAS_HUGE_LOGDET(eye(3))    Calculate log-determinant of 3x3
      identity matrix.
```

tapas_huge_logit

This function is intended for internal use only. Do not call directly

```
Numerical stable calculation of logit function.

INPUTS:
  x - Array of double.

OUTPUTS:
  y - logit of x.
```

tapas_huge_parse_inputs

This function is intended for internal use only. Do not call directly

```
Parse name-value pair type arguments into a struct.

INPUTS:
  opts - Struct containing all valid names as field names and
         corresponding default values as field values.

OPTIONAL INPUTS:
  Name-value pair arguments.

OUTPUTS:
  opts - Struct containing name-value pair input arguments as fields.
```

```
EXAMPLE:
  opts = TAPAS_HUGE_PARSE_INPUTS(struct('a',0,'b',1),'a',10)    Specify
      'a' and 'b' as valid property names and receive non-default value
      for 'a'.
```

7 Licence and Support

This toolbox is part of TAPAS, which is released under the terms of the GNU General Public Licence (GPL), version 3. For further details, see:

`https://www.gnu.org/licenses/`

The software contained in this toolbox is provided "as is", without warranty of any kind, express or implied, including, but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement.

This software is intended for research only. Do not use for clinical purpose. Please note that this toolbox is under active development. Considerable changes may occur in future releases.

Support for this toolbox is provided via the GitHub page of TAPAS. For questions and bug reports, please visit:

`https://github.com/translationalneuromodeling/tapas/issues`

References

Klaas E. Stephan, Nikolaus Weiskopf, Peter M. Drysdale, Peter A. Robinson, and Karl J. Friston. Comparing hemodynamic models with dcm. *NeuroImage*, 38(3):387–401, 2007. ISSN 1053-8119. doi: 10.1016/j.neuroimage.2007.07.040.

Yu Yao, Sudhir S. Raman, Michael Schiek, Alex Leff, Stefan Frässle, and Klaas E. Stephan. Variational bayesian inversion for hierarchical unsupervised generative embedding (huge). *NeuroImage*, 179:604–619, 2018. ISSN 1053-8119. doi: 10.1016/j.neuroimage.2018.06.073.