BDTB 解説資料

Ver. 1.1

2011/08/03

はじめに

本資料は，Brain Decoder Toolbox (BDTB) の解説資料です.
BDTB は，各条件における脳活動パターンの違いを学習し，その結果をもちいて脳活動を分類することにより，脳活動の“デコーディング”を実現します.

BDTB は，Matlab の M-ファイルからなる関数群です.
OS に依存することなく，Matlab 上で使用可能です.
なお，動作確認は，Windows7 Professional，Matlab R2010a の環境で行っております.

BDTB 内の関数には，“SPM5”の関数を使用するものがあります.
また，BDTB は現在，“LIBLINEAR”，“LIBSVM”，“OSU-SVM”，“SLR”を分類器として使用可能です.
同梱されていないツールにつきましては，各自でご用意ください.

・SPM5
http://www.fil.ion.ucl.ac.uk/spm/software/spm5/

・LIBLINEAR
http://www.csie.ntu.edu.tw/~cjlin/liblinear/

・LIBSVM
http://www.csie.ntu.edu.tw/~cjlin/libsvm

・OSU-SVM
http://svm.sourceforge.net/download.shtml

・SLR
http://www.cns.atr.jp/~oyamashi/SLR_WEB.html

著作権および免責事項

本ソフトウェアは GPL に準拠しています.
そのため，本ソフトウェアの複製物を所持している者に，以下のことが許可されています.

1. プログラムの実行
2. プログラムの動作を調べ，それを改変すること
3. プログラムを改良し，改良を公衆にリリースする権利
4. 複製物の再頒布

なお，GPL はコピーレフトのため，GPL でライセンスされた著作物の派生的著作物に関しても GPL でライセンスされなければなりません（※）.

※ 本ソフトウェアに限れば，改変済みのソースコードを再頒布する際に，全てのソースコードが改変可能な状態で再頒布されることに該当します.

処理の流れ

BDTB を用いて脳活動をデコーディングする処理の流れは，以下の通りです.

1. Mat ファイル作成
   a. 実験デザインの読み込み
   b. 脳情報の読み込み
   c. Mat ファイルへの書き出し

2. デコーディング
   a. データの前処理
   b. モデルの学習
   c. データの分類

Mat ファイル作成

脳情報や実験デザインを Matlab に読み込み, BDTB で処理可能な構造体に格納したうえで, Mat ファイルに書き出します.

BDTB で想定している構造体の仕様は，以下の通りです.

D 構造体（データ）

フィールド名	内容	フォーマット
data	脳情報	[時間(sample)×空間(ch／vox)] の実数
label	提示刺激や運動の種類などの条件	[時間×ラベル種類] の整数
label_type	条件の種類	{1×ラベル種類} の文字列
label_def	各条件の名前	{1×ラベル種類}{1×条件数} の文字列
design	何番目の run／session／block などの実験デザイン	[時間×デザイン種類] の整数
design_type	各実験デザインの名前	{1×デザイン種類} の文字列
stat	各サンプルの統計量	[統計量種類×空間] の実数
stat_type	各統計量の名前	{1×統計量種類} の文字列
xyz	ch／vox の座標値	[3(x,y,z)×空間] の実数
roi	ch／vox が ROI(Region Of Interest) に含まれているか否か	[ROI 種類×空間] の 0／1
roi_name	各 ROI の名前	{1×ROI 種類} の文字列

デコーディング

作成した Mat ファイルから D 構造体を Matlab に読み込み, トレーニング用データとテスト用データに分け,

・トレーニング用データで，脳情報とラベルの対応関係を学習

・テスト用データで，学習モデルによるラベルの推定

を行います.

BDTB には，クロスバリデーションによりテストを行う関数や，データに適用する各種フィルタなどが用意されています.

これらを使用するためには，Mat ファイル作成 にて説明しました D 構造体に脳情報や実験デザインを格納し，各関数のパラメータを P 構造体に格納し，各関数に引数として渡す必要があります.

BDTB で想定している P 構造体の仕様は，以下の通りです.

フィールド名	内容	フォーマット
<関数名>．<パラメータ名>	パラメータ値	関数名ごとの構造体
（例）		
selectChanByTvals.num_chans	200	
selectChanByTvals.tvals_min	3.2	
reduceOutliers.std_thres	4	

※ 各関数のパラメータにつきましては，関数一覧 や，各関数のヘルプを参照ください.

サンプルプログラム

web サイト(http://www.cns.atr.jp/dni/download/brain-decoder-toolbox/)からダウンロード可能なサンプルプログラムの解説を行います.

サンプルプログラムでは，実験データとして fMRI データを用い，上記 処理の流れ に従って，Mat ファイルの作成 ，デコーディング を行います.

サンプルデータの実験内容は，以下の通りです.

被験者：1 名
MRI 設定：1.5 T MRI(Shimadzu-Marconi)，FOV 192 mm，64×64 matrix，3×3×3 mm，50 slices，TE 50 ms，FA 90 °，TR 5 s

MRI 内の被験者に，ジャンケンの手の形を作ってもらった.
どの形を作るかは画像を提示して指示した.
1 秒間隔の音声指示に従い, 脱力状態から指示された形へと手の変形を繰り返させた.
実験開始から 20 秒間の rest 期間の後，20 秒ごとに手の形の指示を変更し， 最後に再び 20 秒間の rest をもって，1 run とする.
手の形の指示は，次の順序で行った.

Run 1：	rest	チョキ	グー	パー	グー	パー	チョキ	rest
Run 2：	rest	チョキ	パー	グー	グー	チョキ	パー	rest
Run 3：	rest	グー	チョキ	パー	グー	パー	チョキ	rest
Run 4：	rest	パー	グー	チョキ	グー	チョキ	パー	rest
Run 5：	rest	チョキ	パー	グー	グー	チョキ	パー	rest
Run 6：	rest	パー	チョキ	グー	パー	グー	チョキ	rest
Run 7：	rest	パー	グー	チョキ	グー	パー	チョキ	rest
Run 8：	rest	チョキ	パー	グー	チョキ	グー	パー	rest
Run 9：	rest	チョキ	グー	パー	グー	パー	チョキ	rest
Run10：	rest	グー	チョキ	パー	チョキ	パー	グー	rest

make_fmri_mat.m

この関数は，脳情報と実験デザインを読み込み，D 構造体に格納したうえで，Mat ファイルを作成します.
D 構造体につきましては，Mat ファイル作成 を参照ください.

各設定項目は，以下のとおりです.

P.sbj_id = 'SS100511'

被験者 ID. 大文字イニシャルと実験日時（YYMMDD フォーマット）で指定.

P.paths.to_lib = ''
P.paths.to_dat = ''

BDTB／実験データのルートディレクトリへのパスを指定.
未設定の場合，スクリプト実行時に，ダイアログにて指定.

P.fMRI.TR = 5

MRI 計測の TR を指定.

P.fMRI.begin_vols = 3

各 run の最初のファイル番号を指定(サンプルでは,2 ファイル削除しているので 3).
run ごとに異なる場合は，[1×run 数] のリストで指定.

P.fMRI.run_names = { 'a' ,' b' ,' c' ,' d' ,' e' ,' f' ,' g' ,' h' ,' i',' j' }

ファイル名に含まれる，run 数を示す文字を指定.（※１）

P.fMRI.base_file_name = ['r' P.sbj_id]

ファイルのベースとなる名前を指定.（※１）

P.prtcl.labels_runs_blocks = ... （1：rest，2：グー，3：チョキ，4：パー）

ブロックごとの条件／ラベル（整数）を，run ごとにセルにまとめ，ラベルの種類ごとにセルを分けて指定.（※２）

P.prtcl.labels_type = { 'rock-paper-scissors' }

条件／ラベルの種類を，セル配列の文字列で指定.

P.prtcl.labels_def　= {‘rest’,’ rock’,’ scissors’,’ paper’}

各条件／ラベルの名前を，ラベルの種類ごとに，セル配列の文字列で指定.

P.prtcl.samples_per_label　= {4}

Labels_runs_blocks で指定した各ラベルに割り当てるサンプル数を，ラベルの種類ごとに指定.

全 run，全ラベルで等しい場合は[1×1]の数字で指定，ラベルごとに異なるが run では同じ場合は[1×ラベル数]で指定，run でも異なる場合は全 run 分を指定.（※２）

P.prtcl.samples_per_block　= 4

ブロックごとのサンプル数を指定.

ラベルとブロックが一致する場合は，samples_per_label と同じ値を指定.

P.rois.spm_ver　= 5

ROI 作成に使用した SPM のバージョンを指定.

（SPM のバージョンにより，左右の定義が変わるため）

P.rois.roi_set　= ‘roi’

ROI の名前を指定（作成する Mat ファイル名に使用）.

P.rois.roi_dir　= ‘roi/’

ROI ファイルがあるディレクトリ名を指定.

P.rois.roi_files　= {‘M1_RHand’,’ SMA_RHand’,’ CB_RHand’,...}

ROI ファイル名を，セル配列の文字列で指定（拡張子.mat は不要）.

ROI を使用しない場合は，空のセル配列を指定.

P.stats.stat_dir　= ‘roi/’

統計量が保存されたファイルがあるディレクトリ名を指定

P.stats.stat_files　= {{‘VOX_CB_RHand.mat’,’ VOX_M1_RHand.mat’,...}}

統計量が保存されたファイル名を，セル配列の文字列で指定.

複数ファイルの統計量を統合する場合は，セルにまとめて指定.

P.stats.stat_type　= {‘tval’}

統計量の名前を，セル配列の文字列で指定.

P.output.verbose　= 0

スクリプト実行時のメッセージ出力レベルを指定.

0（出力なし）～2（全て出力）

P.output.save_ver　= 7

作成する Mat ファイルのフォーマットを指定.

P.output.file_name　= [P.sbj_id '_fmri_' P.rois.roi_set '_v' …]

作成する Mat ファイル名を指定.

※1　サンプルでは，fMRI ファイル名は，

[接頭文字][被験者 ID][run を表す文字][4 ケタのファイル番号].hdr／.img

となっております.

そのため，run_names には[run を表す文字(a, b, …, j)] を，base_file_name には[接頭文字][被験者 ID] を指定します.

※2　各サンプルに対応するラベル D.label は，labels_runs_blocks と samples_per_label を組み合わせて作成します.

samples_per_label が[1×1]のとき：

labels_runs_blocks の各ラベルを samples_per_label 数だけ繰り返し, label とします

samples_per_label が[1×n]のとき：

labels_runs_blocks の各 run，i 番目のラベルを，samples_per_label(i)数だけ繰り返し，label とします

samples_per_label が{1×m}[1×n]のとき：

labels_runs_blocks の j 番目の run，i 番目のラベルを，samples_per_label{j}(i)数だけ繰り返し，label とします

decode_basic.m

この関数は，Mat ファイル作成 で説明しましたD構造体を読み込み，各種フィルタを適用したのち，クロスバリデーションによってデコーディング精度を求めます.

各設定項目は，以下のとおりです.

P.script_name = mfilename
P.date_time = datestr(now,' yyyy-mm-dd HH:MM:SS')
実行関数名と実行日時を格納.

P.paths.to_lib = ''
P.paths.to_dat = ''
BDTB／実験データのルートディレクトリへのパスを指定.
未設定の場合，スクリプト実行時に，ダイアログにて指定.

P.procs1 = {...}
クロスバリデーション前に適用するフィルタの名前を指定.
指定した順序に適用される．(※３)

P.procs2 = {...}
クロスバリデーション中に適用するフィルタの名前を指定.
指定した順序に適用される．(※３)

P.<関数名>.<パラメータ名> = ...
各フィルタのパラメータを設定．(※４)

P.models = { 'libsvm_bdtb' }
学習モデルを，セル配列の文字列で指定.
複数のモデルを指定したときは，各モデルで並行にデコーディングを行う.

P.<モデル名>.<パラメータ名> = ...
各モデルのパラメータを設定．(※４)

※３　procs1 に指定したフィルタは，クロスバリデーション前に適用されます．
つまり，トレーニング用データとテスト用データに分ける前に適用されますので，トレーニング時には分からないはずのテスト用データの情報が含まれてしまわないよう，注意が必要です（情報漏洩）．
procs2 に指定したフィルタは，クロスバリデーション中に，トレーニング用データとテスト用データに別々に適用します（トレーニング時のパラメータをテスト時に引き継ぐことも可能です）．

※４　各フィルタ，モデルで設定可能なパラメータにつきましては，関数一覧 や，各関数のヘルプを参照ください．

関数一覧

BDTB で提供している関数の一覧です.

フィルタ

averageBlocks	ブロックごとにデータを平均化
averageLabels	ラベルごとにデータを平均化
balanceLabels	ラベルごとのサンプル数の平衡化
convertLabel	ラベルの置換
detrend_bdtb	時間方向のトレンド除去
highPassFilter	ハイパスフィルタ
normByBaseline	ベースラインによる正規化
poolSample	ラベルごとにデータを平均化
reduceOutliers	外れ値の除去
removeBlockSample	block 単位でのデータ削除
selectBlockSample	block 単位でのデータ選択
selectChanByTvals	t 値によるチャンネル選択
selectConds	ラベルによるデータ選択
selectLabelType	ラベル種類の選択
selectTopFvals	F 値によるデータ選択
shiftData	時間方向のデータ移動
zNorm_bdtb	z スコアに正規化

モデル

liblinear_bdtb	liblinear によるデコーディング
libsvm_bdtb	libsvm によるデコーディング
slr_lap_bdtb	SLR-LAP-1vsR によるデコーディング
slr_var_bdtb	SLR-VAR-1vsR によるデコーディング
smlr_bdtb	Multinomial SLR によるデコーディング
svm11lin_bdtb	OSU SVM によるデコーディング

評価

crossValidate	クロスバリデーションの実行
validate	バリデーションの実行

averageBlocks　　ブロックごとにデータを平均化

[D, pars] = averageBlocks(D, pars)

同一ブロック内のデータを 1 つにまとめ，平均化します

Input :

D.data	－ 脳情報
D.label	－ 提示刺激や運動の種類などの条件
D.design	－ 実験デザイン（block 情報を取得）
D.design_type	－ 実験デザインの名前（‘block’を探す）

Optional :

pars.begin_off	－ 平均化に使わず削除する，各ブロックの先頭からのサンプル数（デフォルト：0）
pars.end_off	－ 平均化に使わず削除する，各ブロックの末尾からのサンプル数（デフォルト：0）
pars.target_labels	－ 平均化する label（デフォルト：全ラベル）
pars.verbose	－ メッセージ出力レベル（デフォルト：1）

Output :

D.data	－ 平均化された脳情報
D.label	－ 平均化されたサンプルに合わせたラベル
D.design	－ 平均化されたサンプルに合わせた実験デザイン

averageLabels　　ラベルごとにデータを平均化

[D, pars] = averageLabels(D, pars)

連続するラベルに対応するデータを 1 つにまとめ，平均化します

Input :

D.data	― 脳情報
D.label	― 提示刺激や運動の種類などの条件

Optional :

pars.begin_off	― 平均化に使わず削除する，各ブロックの先頭からのサンプル数（デフォルト：0）
pars.end_off	― 平均化に使わず削除する，各ブロックの末尾からのサンプル数（デフォルト：0）
pars.target_labels	― 平均化する label（デフォルト：全ラベル）
pars.verbose	― メッセージ出力レベル（デフォルト：1）

Output :

D.data	― 平均化された脳情報
D.label	― 平均化されたサンプルに合わせたラベル
D.design	― 平均化されたサンプルに合わせた実験デザイン

balanceLabels　　ラベルごとのサンプル数の平衡化

[D, pars] = balanceLabels(D, pars)

ラベルごとのサンプル数を，同一にそろえます

Input :

D.data	― 脳情報
D.label	― 提示刺激や運動の種類などの条件

Optional :

pars.method	― サンプル数のそろえ方 1：平均にそろえる，2：最小にそろえる（デフォルト）， 3：最大にそろえる
pars.doTest	― テスト時にサンプル数をそろえるか？ 0：no，1：yes（デフォルト）
pars.mode	― 1：トレーニング，2：テスト
pars.verbose	― メッセージ出力レベル（デフォルト：0）

Output :

D.data	― 新しいラベルに合った脳情報
D.label	― サンプル数がそろえられたラベル

convertLabel　　ラベルの置換

[D, pars] = convertLabel(D, pars)

与えられた置換テーブルに従い，ラベルを置換します

Input：

D.label　　－ 提示刺激や運動の種類などの条件

pars.list　　－ 置換テーブル

{[元ラベル 1, 新ラベル 1], [元ラベル 2, 新ラベル 2], …} フォーマット

Output：

D.label　　－ 置換されたラベル

detrend_bdtb　　時間方向のトレンド除去

[D, pars] = detrend_bdtb(D, pars)

時間方向のトレンドを除去します

Input :

D.data	― 脳情報

Optional :

D.design	― 実験デザイン（run 情報を取得）
D.design_type	― 実験デザインの名前（‘run’を探す）
pars.sub_mean	― 平均値を引くか？ 0：no（デフォルト），1：yes
pars.method	― 除去手法 linear：線形トレンドの除去（デフォルト）， constant：平均値の除去
pars.breaks	― 時間の区分（サンプル）を指定 [開始点 1, 開始点 2, …; 終了点 1, 終了点 2, …]フォーマット
pars.break_run	― run 情報を時間の区分として使うか？ 0：no，1：yes（デフォルト）
pars.verbose	― メッセージ出力レベル（デフォルト：1）

Output :

D.data	― トレンド除去された脳情報

※ 時間方向のトレンド除去のため，時間が連続でなければ正しく働きません.
時間の区分を pars.breaks で指定するか，D.design から run 情報を取得する必要があります.

highPassFilter　　ハイパスフィルタ

[D, pars] = highPassFilter(D, P)

ハイパスフィルタを適用します

Input :

D.data	－ 脳情報

Optional :

D.design	－ 実験デザイン（run 情報を取得）
D.design_type	－ 実験デザインの名前（‘run’を探す）
pars.dt	－ サンプリング間隔 [sec]（デフォルト：2）
pars.cutoff	－ カットオフ周波数 [sec]（デフォルト：128） 区分ごとに変える場合はリストで指定（[128, 128, …]）
pars.app_dim	－ 適用次元 1：時間方向（デフォルト），2：空間方向
pars.linear_detrend	－ トレンド除去を前に実行するか？ 0：no，1：yes（デフォルト）
pars.breaks	－ 時間の区分（サンプル）を指定 [開始点 1, 開始点 2, …; 終了点 1, 終了点 2, …]フォーマット
pars.break_run	－ run 情報を時間の区分として使うか？ 0：no，1：yes（デフォルト）
pars.verbose	－ メッセージ出力レベル（デフォルト：1）

Output :

D.data	－ フィルタを適用された脳情報

※ 時間方向に適用する場合は，時間が連続でなければ正しく働きません.
時間の区分を pars.breaks で指定するか，D.design から run 情報を取得する必要があります.

normByBaseline　ベースラインによる正規化

[D, pars] = normByBaseline(D, pars)

時間方向のベースラインを求め，その値で正規化します

Input：

D.data	― 脳情報
D.label	― 提示刺激や運動の種類などの条件

Optional：

D.design	― 実験デザイン（run 情報を取得）
D.design_type	― 実験デザインの名前（‘run’を探す）
pars.base_conds	― ベースラインの導出に使用するラベル（デフォルト：1）
pars.zero_thres	― ベースラインを 0 とみなす閾値（デフォルト：1）
pars.breaks	― 時間の区分（サンプル）を指定 [開始点 1, 開始点 2, …; 終了点 1, 終了点 2, …]フォーマット
pars.break_run	― run 情報を時間の区分として使うか？ 0：no，1：yes（デフォルト）
pars.verbose	― メッセージ出力レベル（デフォルト：1）
pars.mode	― 正規化手法 0: 平均を引き, 平均で割る (% signal change, デフォルト)， 1：平均で割る，2：平均を引く， 3：平均を引き，標準偏差で割る（z score）

Output：

D.data	― 正規化された脳情報

※ 時間方向に正規化するため，時間が連続でなければ正しく働きません.
時間の区分を pars.breaks で指定するか，D.design から run 情報を取得する必要があります.

poolSample　　ラベルごとにデータを平均化

[D, pars] = poolSample(D, pars)

同一ラベルのデータをまとめ，平均化します
パラメータ設定により，異なる block（不連続）のラベルの平均化にも対応します

Input :

D.data	― 脳情報
D.label	― 提示刺激や運動の種類などの条件
D.design	― 実験デザイン（block，run 情報を取得）
D.design_type	― 実験デザインの名前（‘block’，‘run’を探す）
pars.nPool	― まとめるサンプル数
pars.poolLabel	― まとめるラベル

Optional :

pars.poolSep	― 異なる block のサンプルもまとめるか？ 0：no，1：yes（デフォルト）
pars.useResid	― 最後に余ったサンプルをどうするか？ 0：削除，1：最後のグループに追加（デフォルト）， 2：新たなグループとして追加

Output :

D.data	― 平均化された脳情報
D.label	― まとめられたラベル
D.design	― まとめられたラベルに合わせた実験デザイン

※ データをまとめるにあたり，実験デザインを作り直します.
（各 run において，非対象ラベルを先に，まとめたラベルをその後ろに）
時間方向の順序が変わりますので，適用タイミングにはご注意ください.

reduceOutliers　　外れ値の除去

[D, pars] = reduceOutliers(D, pars)

時間／空間方向の外れ値を除去します

Input :

D.data	― 脳情報

Optional :

D.design	― 実験デザイン（run 情報を取得）
D.design_type	― 実験デザインの名前（‘run’を探す）
pars.app_dim	― 適用次元 1：時間方向（デフォルト），2：空間方向
pars.remove	― 外れ値を含むチャンネルを削除するか？ 0：値の修正のみ，1：チャンネルを削除
pars.method	― 外れ値の判定方法 1：最大標準偏差のみ，2：指定した最大値・最小値のみ， 3：両方（デフォルト）
pars.std_thress	― 閾値を標準偏差の何倍にするか（デフォルト：3）
pars.num_its	― 繰り返し回数（デフォルト：10）
pars.max_val	― 最大値の指定（デフォルト：inf）
pars.min_val	― 最小値の指定（デフォルト：-inf）
pars.breaks	― 時間の区分（サンプル）を指定 [開始点 1, 開始点 2, …; 終了点 1, 終了点 2, …]フォーマット
pars.break_run	― run 情報を時間の区分として使うか？ 0：no，1：yes（デフォルト）
pars.verbose	― メッセージ出力レベル（デフォルト：1）

Output :

D.data	― 外れ値が除去された脳情報
D.xyz	― 削除されなかったチャンネルの XYZ 座標値
D.stat	― 削除されなかったチャンネルの統計量
D.roi	― 削除されなかったチャンネルの ROI 情報

※ 時間方向の処理のため，時間が連続でなければ正しく働きません.
時間の区分を pars.breaks で指定するか，D.design から run 情報を取得する必要があります.

※ チャンネル削除を選択した場合，削除されなかったチャンネルに従って，xyz，stat，roi の内容が更新されます.

removeBlockSample　　block 単位でのデータ削除

[D, pars] = removeBlockSample(D, pars)

block ごとに，指定した数のデータを削除します

Input :

D.data	－ 脳情報
D.label	－ 提示刺激や運動の種類などの条件
D.design	－ 実験デザイン（block 情報を取得）
D.design_type	－ 実験デザインの名前（‘block’を探す）
pars.begin_off	－ block の先頭からの，削除するサンプル数（デフォルト：0）
pars.end_off	－ block の末尾からの，削除するサンプル数（デフォルト：0）

Optional :

pars.target_labels	－ 処理対象のラベル（デフォルト：全ラベル）
pars.verbose	－ メッセージ出力レベル（デフォルト：1）

Output :

D.data	－ 削除された脳情報
D.label	－ 削除された脳情報に合わせたラベル
D.design	－ 削除された脳情報に合わせた実験デザイン

selectBlockSample　　block 単位でのデータ選択

[D, pars] = selectBlockSample(D, pars)

block ごとに，指定した数のデータを選択します

Input :

D.data	― 脳情報
D.label	― 提示刺激や運動の種類などの条件
D.desing	― 実験デザイン（block 情報を取得）
D.design_type	― 実験デザインの名前（‘block’を探す）
pars.inds	― 選択するサンプルのインデックス

Optional :

pars.target_labels	― 処理対象のラベル（デフォルト：全ラベル）
pars.verbose	―メッセージ出力レベル（デフォルト：1）

Output :

D.data	― 選択された脳情報
D.label	― 選択された脳情報に合わせたラベル
D.design	― 選択された脳情報に合わせた実験デザイン

selectChanByTvals　　t 値によるチャンネル選択

[D, pars] = selectChanByTvals(D, pars)

t 値が指定した範囲内にあるチャンネルから，指定した数／割合を選択します

Input：

D.data	－ 脳情報
D.stat	－ 統計量（t 値を取得）
D.stat_type	－ 統計量の名前（‘tval’を探す）

Optional：

pars.num_chans	－ 選択するチャンネル数（整数），または割合（1 以下の小数） （デフォルト：全チャンネル）
pars.tvals_min	－ t 値の最小値（デフォルト：-inf）
pars.tvals_max	－ t 値の最大値（デフォルト：inf）
pars.verbose	－ メッセージ出力レベル（デフォルト：1）

Output：

D.data	－ 選択された脳情報
D.xyz	－ 選択されたチャンネルの XYZ 座標
D.stat	－ 選択されたチャンネルの統計量
D.roi	－ 選択されたチャンネルの ROI 情報

selectConds　　ラベルによるデータ選択

[D, pars] = selectConds(D, pars)

指定したラベルに対応するデータを選択します

Input :

D.data	― 脳情報
D.label	― 提示刺激や運動の種類などの条件
pars.conds	― 選択するラベル

Optional :

pars.verbose	― メッセージ出力レベル（デフォルト：1）

Output :

D.data	― 選択された脳情報
D.label	― 選択されたラベル

selectLabelType　　ラベル種類の選択

[D, pars] = selectLabelType(D, pars)

複数のラベル種類から，解析対象とするラベルを 1 つだけ選択します

Input :

D.label	― 提示刺激や運動の種類などの条件
pars.target	― 選択するラベル種類のインデックス（整数）

Output :

D.label	― 選択されたラベル
D.label_type	― 選択されたラベル種類の名前
D.label_def	― 選択されたラベルの条件名

※ D 構造体には複数のラベル種類が格納可能ですが，デコーディング解析には，トレーニング・テストに使用するラベルを 1 種類のみ選択する必要があります.

selectTopFvals　　F 値によるデータ選択

[D, pars] = selectTopFvals(D, pars)

F 値を計算し，指定した範囲内にあるデータから，指定した数／割合を選択します

Input：

D.data	－ 脳情報
D.label	－ 提示刺激や運動の種類などの条件

Optional：

pars.inds_fvals	－ F 値で並べられたチャンネルのインデックス （トレーニング時に格納し，テスト時に使用）
pars.fvals	－ 求まった F 値（降順）
pars.mode	－ 1：トレーニング（F 値を計算，データを選択）， 2：テスト（inds_fvals でデータを選択）
pars.app_dim	－ 適用次元 1：時間方向，2：空間方向（デフォルト）
pars.num_comp	－ 選択するデータ数（整数），または割合（1 以下の小数） （デフォルト：すべて）
pars.fvals_min	－ F 値の最小値（デフォルト：-inf）
pars.fvals_max	－ F 値の最大値（デフォルト：inf）
pars.verbose	－ メッセージ出力レベル（デフォルト：1）

Output：

D.data	－ 選択された脳情報

※ F 値は解析データから導出されるため，トレーニング用とテスト用を分ける前に計算を行うと，テスト用データの情報が含まれてしまいます（情報漏洩）.
F 値の導出はトレーニング用データのみで行い，選択するデータを決めたのち，そのインデックスに従ってテスト用データを処理する必要があります.

shiftData　時間方向のデータ移動

[D, pars] = shiftData(D, pars)

データを時間方向に移動します（データとラベルの対応をずらします）

Input :

D.data	－ 脳情報
D.design	－ 実験デザイン（block，run 情報を取得）
D.design_type	－ 実験デザインの名前（‘block’，‘run’を探す）
pars.shift	－ データの移動量（正の整数）

Optional :

pars.verbose	－ メッセージ出力レベル（デフォルト：1）

Output :

D.data	－ 移動された脳情報
D.label	－ 移動された脳情報に合わせたラベル
D.design	－ 移動された脳情報に合わせた実験デザイン

※ fMRI データの解析では，血流変化による遅延を考慮する必要があります.
そのため，脳情報をラベルに対してずらす処理を行います.

※ 時間方向の処理のため，時間が連続でなければ正しく働きません.
時間の区分を D.design から取得します.

zNorm_bdtb　　z スコアに正規化

[D, pars] = zNorm_bdtb(D, pars)

z スコアへの正規化を行います

Input :

D.data	－ 脳情報

Optional :

pars.mode	－ 1：トレーニング（平均・標準偏差の計算）， 2：テスト（平均・標準偏差の使用）
pars.smode	－ 計算モードか？（mode を上書き） 0：no（デフォルト），1：yes
pars.app_dim	－ 適用次元 1：時間方向，2：空間方向（デフォルト）
pars.sub_mean	－ 平均を引くか？ 1：yes（デフォルト），2：no
pars.verbose	－ メッセージ出力レベル（デフォルト：0）
pars.mu	－ 平均（トレーニング時に格納し，テスト時に使用）
pars.sd	－ 標準偏差（トレーニング時に格納し，テスト時に使用）

Output :

D.data	－ 正規化された脳情報

liblinear_bdtb　　liblinear によるデコーディング

[result, pars] = liblinear_bdtb(D, pars)

liblinear を用いてデコーディングを行います

Input :

D.data	－ 脳情報
D.label	－ 提示刺激や運動の種類などの条件

Optional :

pars.model	－ 学習結果（トレーニング時に格納し，テスト時に使用）
pars.mode	－ 1：トレーニング，2：テスト
pars.verbose	－ メッセージ出力レベル（デフォルト：0）
pars.ops	－ “liblinear”のオプション設定 詳細は，liblinear の README を参照

Output :

result.model	－ ‘liblinear_bdtb’
result.pred	－ 予測されたラベル
result.label	－ 正解のラベル
result.weight	－ ウェイトとバイアス

libsvm_bdtb　　libsvm によるデコーディング

[result, pars] = libsvm_bdtb(D, pars)

libsvm を用いてデコーディングを行います

Input :

D.data	－ 脳情報
D.label	－ 提示刺激や運動の種類などの条件

Optional :

pars.model	－ 学習結果（トレーニング時に格納し，テスト時に使用）
pars.mode	－ 1：トレーニング，2：テスト
pars.verbose	－ メッセージ出力レベル（デフォルト：0）

LIBSVM pars :　－ “libsvm”のオプション設定

pars.kernel　　詳細は，libsvm の README を参照

pars.cost

pars.gamma

pars.coef

pars.degree

pars.prob

Output :

result.model	－ ‘libsbm_bdtb’
result.pred	－ 予測されたラベル
result.label	－ 正解のラベル
result.dec_val	－ 判別予測値
result.weight	－ ウェイトとバイアス

slr_lap_bdtb　　SLR-LAP-1vsR によるデコーディング

[result, pars] = slr_lap_bdtb(D, pars)

SLR-LAP（with Laplace approximation）-1vsR を用いてデコーディングを行います

Input：

D.data	－ 脳情報
D.label	－ 提示刺激や運動の種類などの条件

Optional：

pars.conds	－ 対象ラベル
pars.mode	－ 1：トレーニング，2：テスト
pars.verbose	－ メッセージ出力レベル（デフォルト：0）

SLR pars：　－ “SLR”のオプション設定

　pars.scale_mode　　詳細は，SLR の README を参照

　pars.mean_mode

SLR pars for test：

　pars.weight

　pars.ix_eff

　pars.norm_scale

　pars.norm_base

　pars.norm_sep

SLR pars for train：

　pars.nlearn

　pars.ax0

　pars.amax

Output：

result.model	－ ‘slr_lap_bdtb’
result.pred	－ 予測されたラベル
result.label	－ 正解のラベル
result.dec_val	－ 判別予測値
result.weight	－ ウェイトとバイアス

slr_var_bdtb　　SLR-VAR-1vsR によるデコーディング

[result, pars] = slr_var_bdtb(D, pars)

SLR-VAR（with variational approximation）-1vsR を用いてデコーディングを行います

Input：

D.data	― 脳情報
D.label	― 提示刺激や運動の種類などの条件

Optional：

pars.conds	― 対象ラベル
pars.mode	― 1：トレーニング，2：テスト
pars.verbose	― メッセージ出力レベル（デフォルト：0）

SLR pars：　― “SLR”のオプション設定

pars.scale_mode　　詳細は，SLR の README を参照

pars.mean_mode

SLR pars for test：

pars.weight

pars.ix_eff

pars.norm_scale

pars.norm_base

pars.norm_sep

SLR pars for train：

pars.nlearn

pars.ax0

pars.amax

Output：

result.model	― ‘slr_var_bdtb’
result.pred	― 予測されたラベル
result.label	― 正解のラベル
result.dec_val	― 判別予測値
result.weight	― ウェイトとバイアス

smlr_bdtb　　Multinomial SLR によるデコーディング

[result, pars] = smlr_bdtb(D, pars)

Multinomial SLR を用いてデコーディングを行います

Input：

D.data	－ 脳情報
D.label	－ 提示刺激や運動の種類などの条件

Optional：

pars.conds	－ 対象ラベル
pars.mode	－ 1：トレーニング，2：テスト
pars.verbose	－ メッセージ出力レベル（デフォルト：0）

SLR pars：　－ “SLR”のオプション設定

　pars.scale_mode　　詳細は，SLR の README を参照

　pars.mean_mode

SLR pars for test：

　pars.weight

　pars.ix_eff

　pars.norm_scale

　pars.norm_base

　pars.norm_sep

SLR pars for train：

　pars.nlearn

　pars.ax0

　pars.amax

Output：

result.model	－ ‘smlr_bdtb’
result.pred	－ 予測されたラベル
result.label	－ 正解のラベル
result.dec_val	－ 判別予測値
result.weight	－ ウェイトとバイアス

svm11lin_bdtb　　OSU SVM によるデコーディング

[result, pars] = svm11lin_bdtb(D, pars)

OSU SVM を用いてデコーディングを行います

Input :

D.data	－ 脳情報
D.label	－ 提示刺激や運動の種類などの条件

Optional

pars.weight	－ 計算したウェイト（トレーニングで格納，テストで使用）
pars.mode	－ 1：トレーニング，2：テスト
pars.num_boot	－ ブートストラップサンプル数 0:ブートストラップしない(デフォルト)，正:サンプル数， 負：-num_boot×length(label)をサンプル数に
pars.verbose	－ メッセージ出力レベル（デフォルト：0）

Output :

result.model	－ ‘svm11lin_bdtb’
result.pred	－ 予測されたラベル
result.label	－ 正解のラベル
result.dec_val	－ 判別予測値
result.weight	－ ウェイトとバイアス

※ 配布されている mex ファイルが 32bit 用のため，32bit 環境でのみ使用可能です.

crossValidate　　クロスバリデーションの実行

[result, P] = crossValidate(D, P, procs, models)

leave-one-out クロスバリデーションを実行します

Input :

D.data	― 脳情報
D.label	― 提示刺激や運動の種類などの条件
D.design	― 実験デザイン（データのグループ分けの基準を取得）
procs	― 適用する関数名の配列
models	― 使用するモデル名の配列

Optional :

P.<function>	― procs や models に指定した処理のパラメータ
P.crossValidate.fold_ind	― 実験デザインの何番目の情報をグループ分けに使用するか（デフォルト：1）
P.crossValidate.res_train	― トレーニング結果も返すか？ 0：no（デフォルト），1：yes
P.crossValidate.verbose	― メッセージ出力レベル（デフォルト：1）

Output :

result{}.model	― 使用したモデル名
result{}.pred	― 予測されたラベル
result{}.label	― 正解のラベル
result{}.dec_val	― 判別予測値
result{}.weight	― ウェイトとバイアス
result{}.freq_table	― 度数分布表
result{}.correct_per	― 正答率

※ fold_ind に指定した実験デザインによってデータをグループに分け，クロスバリデーションを行います.
run を指定すれば leave-‘one run’-out に，block を指定すれば leave-‘one block’-out になります.

validate　　バリデーションの実行

[result, P] = validate(D_tr, D_te, P, procs, models)

トレーニング用データでトレーニングしたモデルを用い，テスト用データを評価します

Input：

D_tr.data	－ トレーニング用脳情報
D_tr.label	－ トレーニング用提示刺激や運動の種類などの条件
D_te.data	－ テスト用脳情報
D_te.label	－ テスト用提示刺激や運動の種類などの条件
procs	－ 適用する関数名の配列
models	－ 使用するモデル名の配列

Optional：

P.<function>	－ procs や models に指定した処理のパラメータ
P.validate.res_train	－ トレーニング結果も返すか？ 0：no（デフォルト），1：yes
P.validate.verbose	－ メッセージ出力レベル（デフォルト：1）

Output：

result{}.model	－ 使用したモデル名
result{}.pred	－ 予測されたラベル
result{}.label	－ 正解のラベル
result{}.dec_val	－ 判別予測値
result{}.weight	－ ウェイトとバイアス
result{}.freq_table	－ 度数分布表
result{}.correct_per	－ 正答率

更新履歴

Ver. 1.1　　2011/08/03
　　間違いを一部訂正

Ver. 1.0　　2011/07/15

連絡先

村田 賢　（むらた さとし）
(株)国際電気通信基礎技術研究所 脳情報研究所 神経情報学研究室
研究技術員
satoshi-m@atr.jp