

MVPANI V1.0 User Manual

Yifan Li, Xiuzhi Wang, Xue Zhou, Qichao Han,
Yanmin Peng, Meng Liang

Tianjin Key Laboratory of Functional Imaging
School of Medical Imaging, Tianjin Medical University

Please cite the following publication:

Yanmin Peng, Xi Zhang, Yifan Li, Sijia Wang, Qian Su, Feng Liu, Chunshui Yu, Meng Liang. MVPANI: a Toolkit with Friendly Graphical User Interface for Multivariate Pattern Analysis of Neuroimaging Data. *Frontiers in Neuroscience*. 2020. 14:545. DOI: 10.3389/fnins.2020.0054

Contents

1 Environmental Requirements.....	1
1.1 Hardware.....	1
1.2 Software	1
2 Installation and Uninstallation	1
2.1 Installation	1
2.2 Uninstallation	1
2.3 Launch MVPANI.....	2
3 Use of MVPANI.....	2
Symbol Description	2
3.1 Main Interface	3
3.1.1 Classification/Regression Specification	3
3.1.2 Save/Load Configurations	3
3.2 Input Module	3
3.2.1 Input Data Files	4
3.2.2 Specify a Mask.....	6
3.2.3 Specify a Label File for All Samples.....	7
3.2.4 Specify a Fold File for Cross-Validation.....	7
3.3 Model Configuration Module.....	9
3.3.1 Specify a Machine Learning Algorithm	9
3.3.2 Data Preprocessing: Feature Normalization.....	12
3.3.3 Data Preprocessing: Feature Dimensionality Reduction.....	12
3.3.4 Data Preprocessing: Feature Selection	13
3.3.5 Searchlight MVPA.....	15
3.4 Output Module	16
3.4.1 The Main Outputs	17

3.4.2 Other Optional Outputs	22
3.4.3 Outputs for Searchlight MVPA.....	26
3.5 Statistical Testing Module	27
3.5.1 Permutation Test Configuration.....	28
3.5.2 Outputs of Permutation Test	29
3.5.3 Outputs of Permutation Test for Searchlight MVPA.....	31
3.6 Data Fusion Module.....	32
3.6.1 Fusion at Feature Level.....	32
3.6.2 Fusion at Decision Level	34
3.7 Exit MVPANI.....	38
4 Examples	38
4.1 Example 1: Classification of Patients and Healthy Controls.....	39
4.1.1 Launch MVPANI and Select Classification.....	39
4.1.2 Data Preparation and Input.....	39
4.1.3 Model Configuration.....	45
4.1.4 Output Specification and Run MVPA	45
4.1.5 Statistical Test to determine the significance of the classification accuracy	49
4.2 Example 2: Classification of Patients and Healthy Controls Using Feature Selection Function.....	51
4.2.1 Launch MVPANI and Select Classification.....	51
4.2.2 Data Preparation and Input.....	52
4.2.3 Model Configuration.....	52
4.2.4 Output Specification and Run MVPA	52
4.2.5 Statistical Test to Determine the Significance of the Classification Accuracy	53
4.3 Example 3: Information Mapping with Searchlight MVPA	54
4.3.1 Data Preparation and Input.....	55
4.3.2 Model Configuration	55
4.3.3 Output	56
4.3.4 Permutation Test	56
4.4 Example 4: Predicting Reaction Time Using Brain Imaging Data.....	57

4.4.1 Launch MVPANI and Select Regression	58
4.4.2 Data Preparation and Input	59
4.4.3 Model Configuration.....	61
4.4.4 Output Specification and Run MVPA	61
4.4.5 Statistical Test to Determine the Significance of the Correlation Coefficient Between the Predicted Values and the True Values	62
4.5 Example 5: Using Data Fusion to Improve Classification Performance	63
4.5.1 Fusion at Feature Level.....	63
4.5.2 Fusion at Decision Level	66

1 Environmental Requirements

1.1 Hardware

The hardware requirements that are recommended are at least: 4-core CPU, 8G RAM, 20G free hard drive space.

1.2 Software

- (1) Runs under Windows Operating System (Windows 7 and above).
- (2) Runs under MATLAB 2015b and above.
- (3) The SPM package (<https://www.fil.ion.ucl.ac.uk/spm/>) needs to be installed in MATLAB.

2 Installation and Uninstallation

2.1 Installation

Once MVPANIV1.0 has been downloaded to your computer, use SetPath in MATLAB to add the entire MVPANIV1.0 folder to the path list as follows:

- (1) Unzip MVPANIV1.0 to a directory on your disk, assuming the directory is: 'F:\MVPA\MVPANI_V1.0' after unzipping;
- (2) Open MATLAB, HOME → SetPath;
- (3) Click 'Add with Subfolders' and select the MVPANIV1.0 folder (i.e. 'F:\MVPA\MVPANI_V1.0');
- (4) Click 'Save'.

2.2 Uninstallation

MVPANIV1.0 can be uninstalled as follows:

- (1) Open MATLAB, HOME → SetPath;
- (2) Select all MVPANIV1.0 folders and subfolders and click 'Remove';
- (3) Click 'Save'.

2.3 Launch MVPANI

Enter 'MVPANI' in the MATLAB Command Window to launch the main interface of MVPANI (Figure 2.3-1).

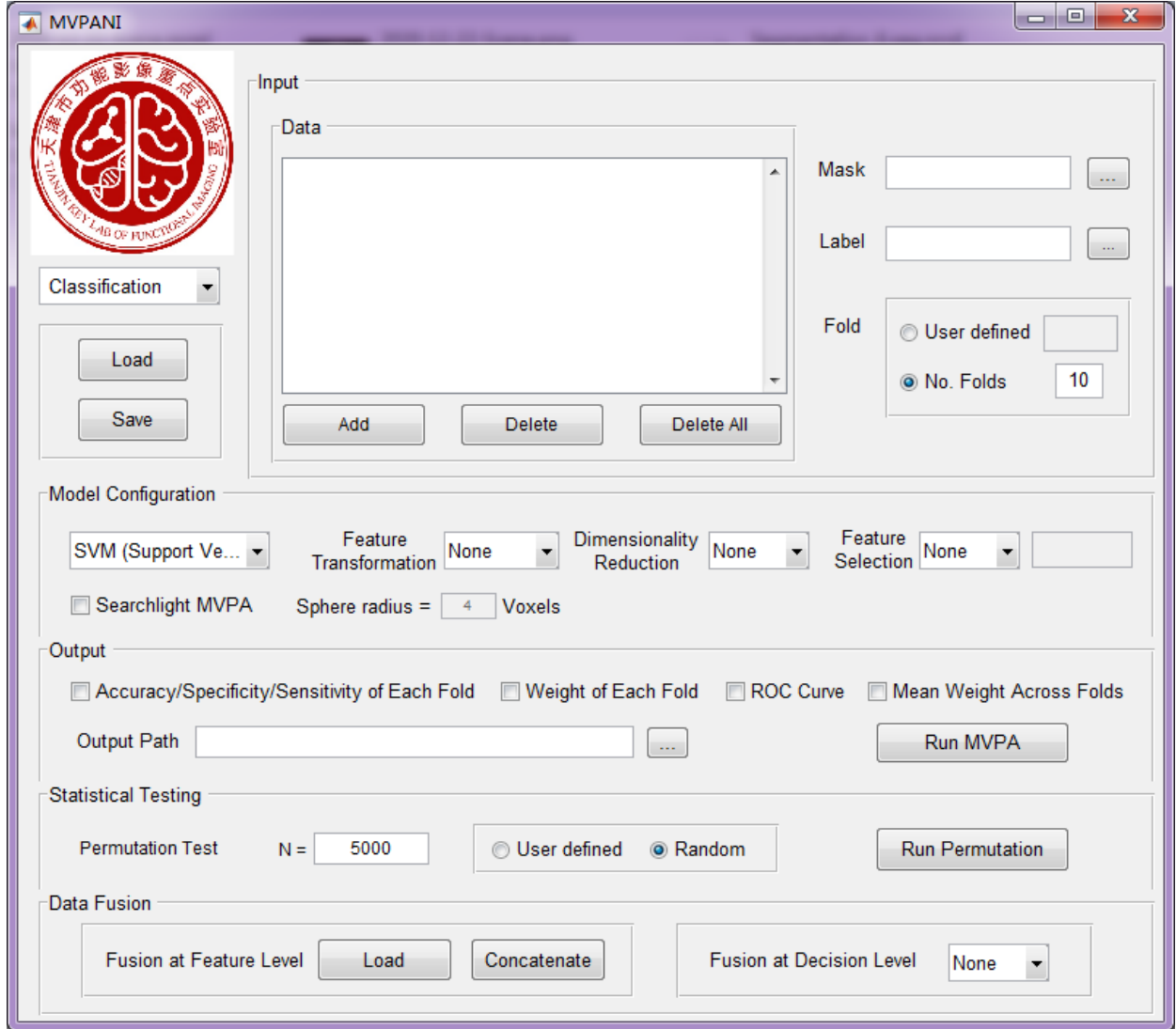


Figure 2.3-1 The main interface of MVPANI

3 Use of MVPANI

Symbol Description

In this chapter, the following symbols are used with specific meanings, and the location where each symbol first appear is also indicated:

n : the number of input samples (Section 3.2.1)

\dot{n} : the \dot{n} th subject (Section 3.2.1)

\dot{j} : the number of features after being filtered by the specified mask (Section 3.2.2)

\dot{l} : the number of classes in a classification task (Section 3.2.3)

\dot{k} : the number of folds specified for cross-validation procedure (Section 3.2.4)

\dot{k} : the \dot{k} th step during cross-validation (Section 3.2.4)

\dot{m} : the number of steps when performing a series of feature selection based on F-scores or feature weights (Section 3.3.4.2)

\dot{m} : a certain step in a series of feature selection (Section 3.3.4.2)

r : the radius of the searchlight sphere in Searchlight MVPA (Section 3.3.5)

N : the number of permutations in a permutation test (Section 3.5.1)

\dot{N} : the \dot{N} th permutation during permutation test (Section 3.5.1)

p : the threshold of P values obtained from permutation test in Searchlight MVPA. (Section 3.5.1)

f : the number of models to be fused in decision fusion (Section 3.6.2.1)

3.1 Main Interface

3.1.1 Classification/Regression Specification

MVPANI supports solving a Classification or a Regression problem, which should to be specified from the dropdown menu at the upper-left part of the main interface (just below the lab logo) after the launch of MVPANI.

3.1.2 Save/Load Configurations

All configurations of a particular analysis can be saved or loaded easily using the "Save" or "Load" button (below the lab logo). You can click the "Save" button to save all the parameters of the whole interface as a mat file, so that you can recall and check them later. Click the "Load" button, and select the parameter file saved with this function in the pop-up dialog box to load the previously saved parameters into the current interface. All configurations are organized in five modules: Input, Model Configuration, Output, Statistical Testing, Data fusion.

3.2 Input Module

This module is for inputting all necessary data files to MVPANI, including the feature data of all samples, the feature mask, the label information of all samples, and the fold information of k-fold cross-validation.

3.2.1 Input Data Files

Supported formats for Data files: nii, img, txt, mat.

This function is used to input the feature data of all samples (we will use n to denote the number of all input samples, and \hat{n} to denote the \hat{n} th sample). Although MVPANI is designed for machine learning analyses of neuroimaging data, it can actually fit all types of data as long as the data can be saved using one of the following data formats: nii, img, txt, mat. Therefore, MVPANI can be used to analyze MRI data (brain maps, connectivity matrices, etc), EEG data (ERP amplitudes/latencies, time series, time-frequency spectrum, etc), clinical data, questionnaires data, etc.

The data of each sample (e.g., each subject) should be saved as a separate data file. Once added, the data of each sample will be expanded into a 1-D vector for subsequent operations (except for the Searchlight MVPA which requires a 2-D or 3-D matrix specifying certain spatial relationships between features, see Section 3.3.6). Note that the data size and structure should be consistent across all samples. The number of input samples is denoted as n .

Click the "Add" button in the Input Module to launch a dialog box (Figure 3.2.1-1) where you can select all data files.

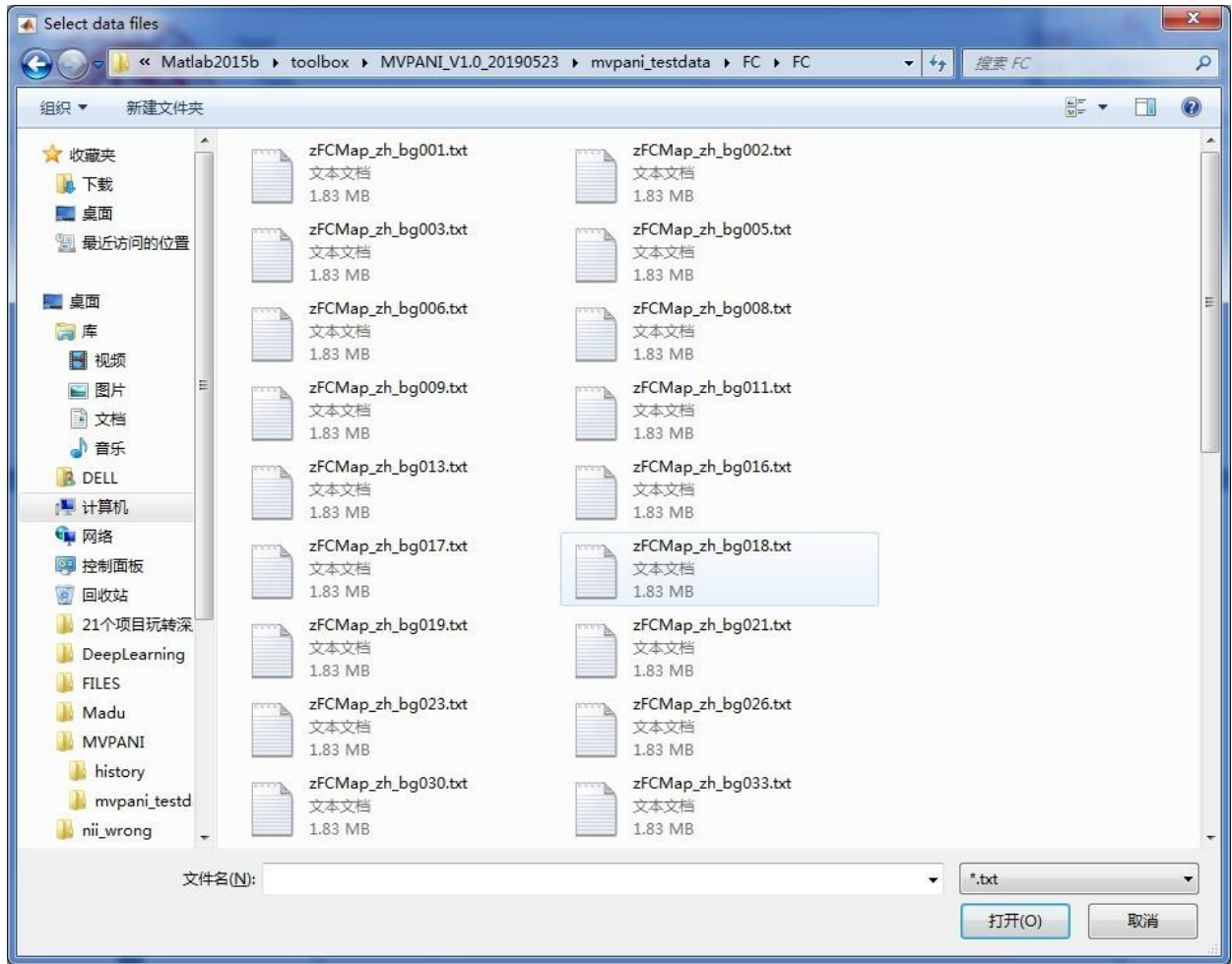


Figure 3.2.1-1 Dialog box for adding sample data files

Select all sample data files and then click "Open" button to read all files in. Note that you need to first change the type of filename extension to match with your file format so that your sample data files will appear in the dialog box; also, all sample files should be selected and loaded in at once, otherwise the interface display and subsequent analyses may be incorrect. You can check the information of all loaded files shown in the Data box to make sure that you have added the correct files (Figure 3.2.1-2).

The order of all loaded sample data files are sorted by filename alphabetically (the order is also shown as ID in the Data box). You need to know the exact order of the sample data files because the Label file and the Fold file (see Sections 3.2.3 and 3.2.4) need to be configured using the same order. It is strongly recommended that you number the files in the filename such as “*_001.nii”, “*_002.nii”, “*_003.nii” and so on, so that the loaded files can be ordered according to some rule you prefer.

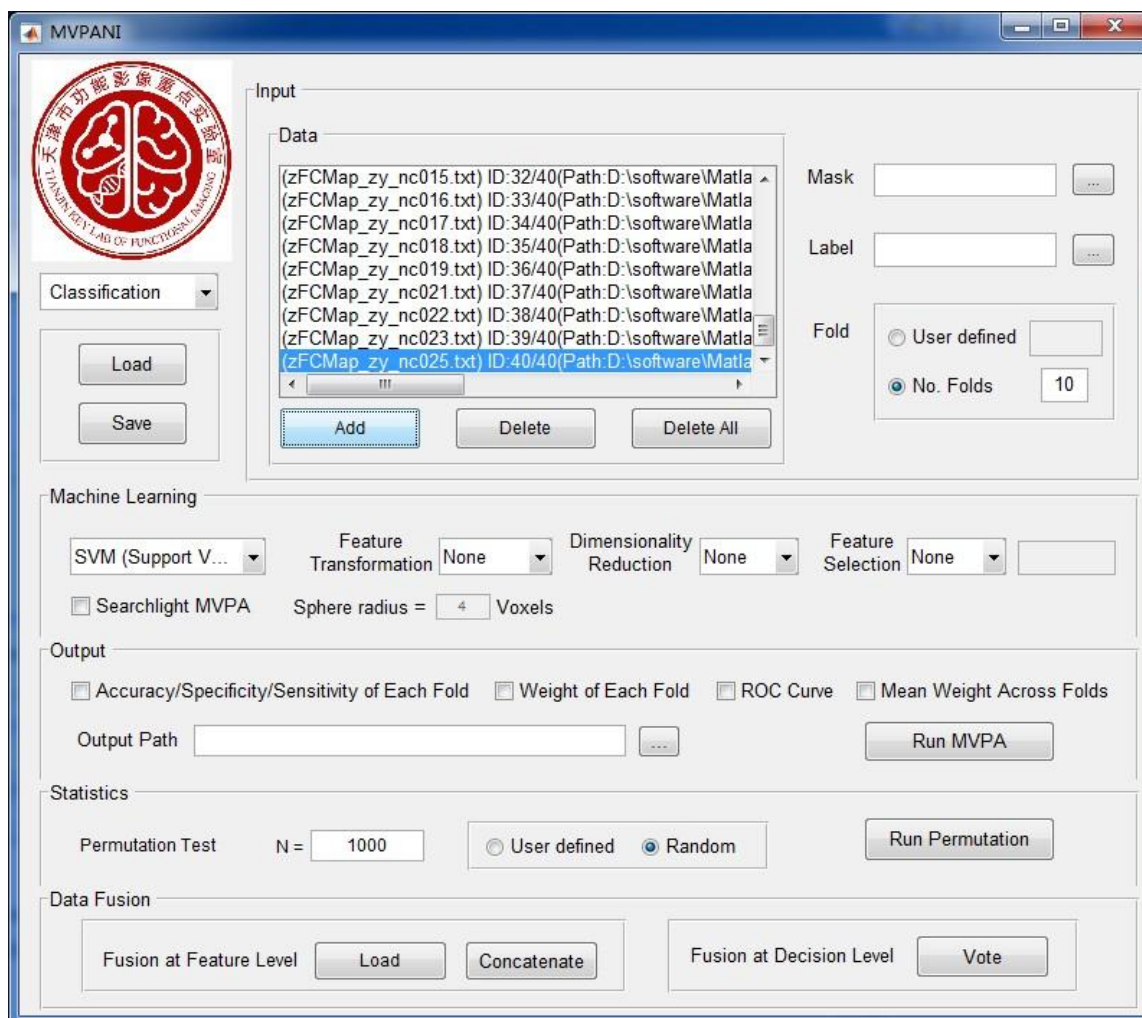


Figure 3.2.1-2 The main interface display after adding all sample data files

Click the "Delete" button to delete the selected files, and click the "Delete All" button to delete all added files.

3.2.2 Specify a Mask

Supported formats for Mask files: nii, img, txt, mat.

Click the "..." button next to the Mask box, you can select a Mask file.

A mask can be specified to indicate which features will be used in the subsequent analyses. The Mask should be a binary matrix (i.e., a matrix with only 0 and 1) with exactly the same structure (i.e., same dimensions, same size, and same order of features) as the input sample data, where 0 indicate the features that will be ignored in the subsequent analyses and 1 indicate the features that will be kept. For example, if the data structure of each sample is a matrix with the size

of $61 \times 73 \times 61$, the Mask must be a binary matrix of the same size of $61 \times 73 \times 61$. We will use j to denote the number of features after being filtered by the specified mask.

In particular, when the sample data (Section 3.2.1) is in .txt or .mat format and you would like to include all features in the subsequent analyses, you do not need to specify a Mask file – leaving the Mask file box blank means that all features in the sample data will be included. Note that, if the sample data is in nii or img format, you have to explicitly specify a Mask file because it is very rare that all voxels in the image will be used in the machine learning model (e.g., at least the voxels outside the brain should be excluded, and in this case, you can specify a whole brain mask with the same size of brain images of each sample).

This function allows you to easily select a subset of the features included in the sample data files, and gives you the flexibility of performing different analyses based on different subsets of all features without changing the original data files.

3.2.3 Specify a Label File for All Samples

Supported formats for Label files: xlsx, xls, mat, txt.

Click the "..." button next to the Label box, you can select a Label file.

The Label file contains the information of the class labels (for classification tasks; e.g. 1/0, 1/-1, or 1/2/3) or the continuous values (for regression tasks) of all input samples. Only numeric labels are supported. Note that, multi-class classifications are only supported for SVM, KNN, RF, DT, and NB (Section 3.2.1), and that ROC curves are not generated for multi-class classifications in the current version of MVPANI. We will use l to denote the number of classes in a classification task.

The class labels or continuous values of all samples should be organized in a row or a column with the length of n (i.e., the number of all samples). Each number in the Label file corresponds to a sample, and the order of the labels must be consistent with the order of loaded sample data files (Section 3.2.1).

3.2.4 Specify a Fold File for Cross-Validation

Supported formats for Fold files: xlsx, xls, mat, txt.

There are two ways to specify a Fold file:

- 1) Default: in this case, you do not need to specify a Fold file but simply specify the number of folds (k ; $2 \leq k \leq$ the number of samples; $k=10$ by default) for cross-validation (just enter the number in the box next to the “No. Folds” text), and MVPANI will

randomly divide all samples into the specified number (e.g., 10) of groups. This corresponds to a k -fold cross-validation.

- 2) If you would like to specify your own Fold file, click the circle next to the “user defined” text and a dialog box will pop up where you can select your Fold file.

To specify your own Fold file, you need to follow a similar rule with setting up a Label file: The Fold file contains the information of which fold each sample belongs to. Only integers (1, 2, 3, ..., k) are allowed and they should be organized in a row or a column with the length of n (i.e., the number of all samples). Each integer in the Fold file corresponds to a sample, and the order of the integers must be consistent with the order of loaded sample data files (Section 3.2.1). For example, if you would like to perform a 3-fold cross-validation procedure on 10 samples, one possible way to specify your Fold file is: '1 2 3 1 2 3 1 2 3 1'. We will use k to denote the k th step during cross-validation procedure.

Up to this point, you have finished the configuration of the Input Module, and the main interface should look like the Figure 3.2.4-1 below.

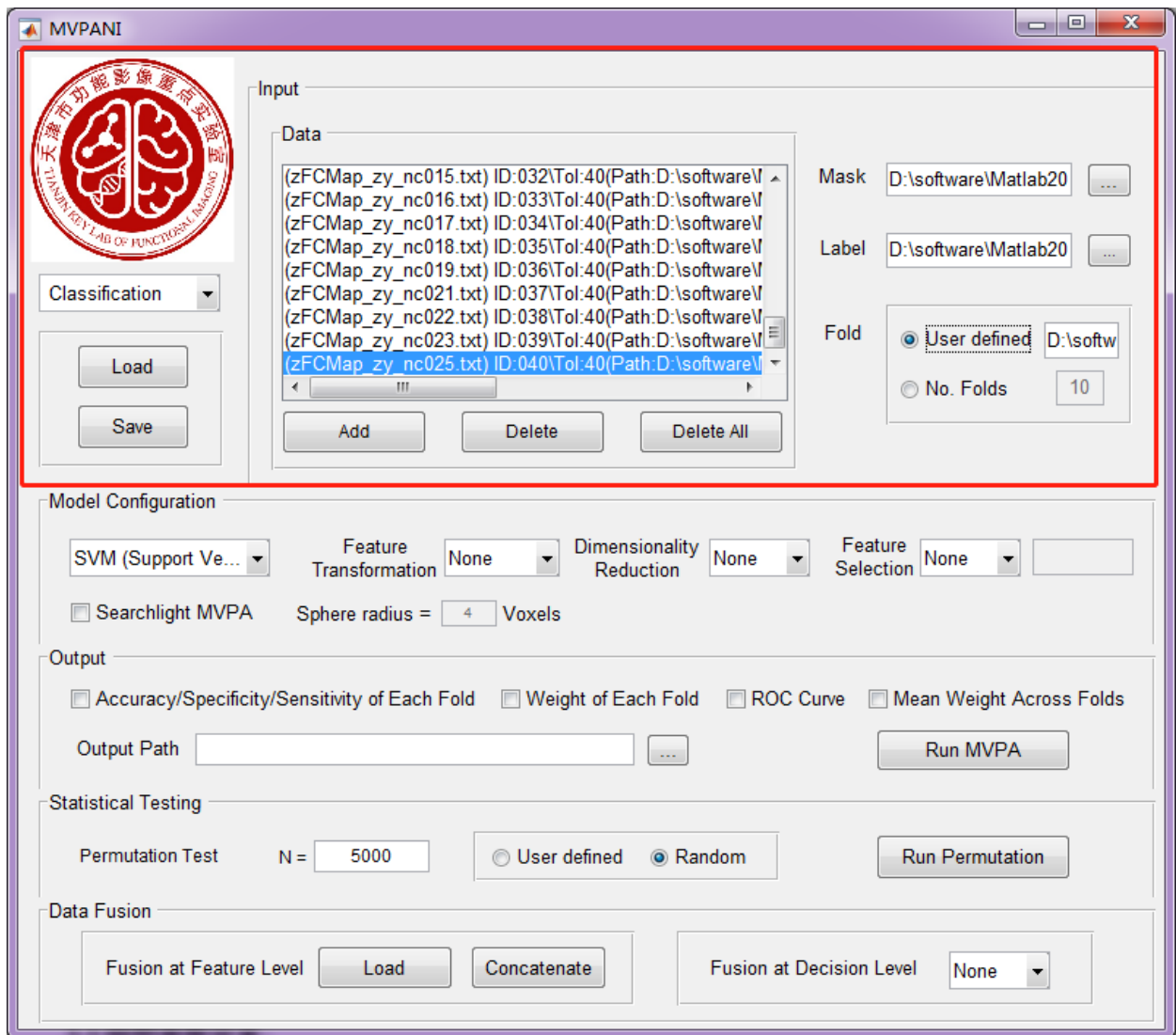


Figure 3.2.4-1 Main interface display after the configuration of the Input Module

3.3 Model Configuration Module

This module is for configuring machine learning algorithms and some optional data preprocessing steps. Note that, as the model configuration for classification tasks is not identical to that for regression tasks, make sure that you redo the model configuration steps if you switched the task type (i.e., classification/regression), otherwise an error may occur.

3.3.1 Specify a Machine Learning Algorithm

The currently implemented machine learning algorithms for classification tasks include Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Random Forest (RF), Decision

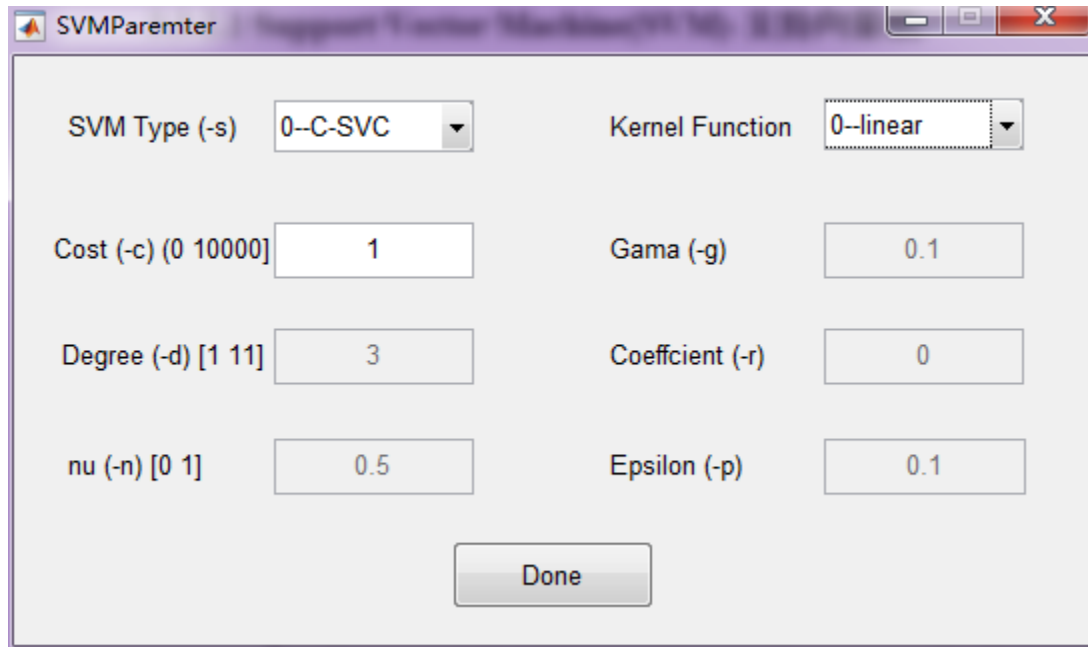
Tree (DT), Linear Discriminate Analysis (LDA), Logistic Regression (LR), and Naive Bayes (NB). The currently implemented machine learning algorithms for regression tasks include SVM, KNN, RF, and DT.

You can specify the machine learning algorithm from the first dropdown menu in the Model Configuration Module.

Note: When using function “Save/Load Configurations”, except for SVM, model hyperparameters are not saved (e.g. KNN, RF) in the current version of MVPANI. Therefore, you must re-enter the model hyperparameters manually when reproducing your results.

3.3.1.1 Support Vector Machine (SVM)/Support Vector Regression (SVR)

This algorithm is named SVM for classification and SVR for regression. To select SVM/SVR, click the corresponding name. A pop-up window will also appear (Figure 3.3.1.1-1) where you can specify the exact model type, kernel function and relevant hyperparameters. Click the "Done" button to confirm your specification. In most cases, you do not need to make changes and the default settings work well.



The image shows a software window titled "SVMParameter" with a standard Windows-style title bar (minimize, maximize, close buttons). The window contains a grid of configuration options for an SVM model. The options are arranged in two columns. The first column includes "SVM Type (-s)" with a dropdown menu set to "0--C-SVC", "Cost (-c) [0 10000]" with a text input field containing "1", "Degree (-d) [1 11]" with a text input field containing "3", and "nu (-n) [0 1]" with a text input field containing "0.5". The second column includes "Kernel Function" with a dropdown menu set to "0--linear", "Gama (-g)" with a text input field containing "0.1", "Coefficient (-r)" with a text input field containing "0", and "Epsilon (-p)" with a text input field containing "0.1". At the bottom center of the window is a "Done" button.

Parameter	Value
SVM Type (-s)	0--C-SVC
Kernel Function	0--linear
Cost (-c) [0 10000]	1
Gama (-g)	0.1
Degree (-d) [1 11]	3
Coefficient (-r)	0
nu (-n) [0 1]	0.5
Epsilon (-p)	0.1

Figure 3.3.1.1-1 The interface for specifying the SVM parameters

Note: The weights are interpretable only for linear SVM/SVR (i.e., when the “Kernel Function” is set to “0--linear”).

3.3.1.2 K-Nearest Neighbor (KNN)

KNN is available for both classification and regression. After clicking the name of this algorithm, a window pops up as shown in Figure 3.3.1.2-1, where you can specify the number of neighbors (for classification, this number is recommended to be an odd number; default value = 11).

Note: there will be no feature weights derived from KNN.

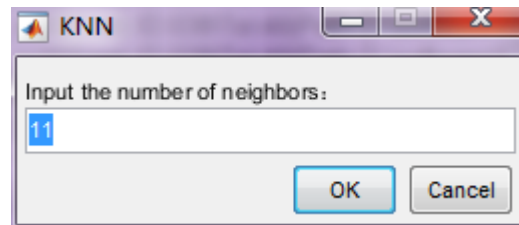


Figure 3.3.1.2-1 The interface for specifying the KNN parameter

3.3.1.3 Random Forest (RF)

RF is available for both classification and regression. After clicking the name of this algorithm, a window pops up as shown in Figure 3.3.1.3-1, where you can specify the number of trees in the random forest (default value = 500).

Note: there will be no feature weights derived from RF.

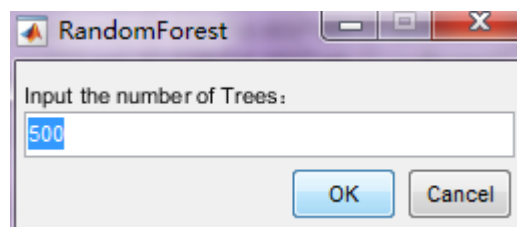


Figure 3.3.1.3-1 The interface for specifying the RF parameter

3.3.1.4 Decision Tree (DT)

DT is available for both classification and regression. To select DT, click the corresponding name. No hyperparameters need to be specified for this algorithm.

Note: there will be no feature weights derived from DT, and ROC curves are not available either.

3.3.1.5 Linear Discriminate Analysis (LDA)

LDA is available only for classification. To select LDA, click the corresponding name. No

hyperparameters need to be specified for this algorithm.

Note: in the current version of MVPANI, there will be no feature weights derived from LDA, and multiclass classification has not been implemented for LDA.

3.3.1.6 Logistic Regression (LR)

LR is available only for classification. To select LR, click the corresponding name. No hyperparameters need to be specified for this algorithm.

Note: in the current version of MVPANI, there will be no feature weights derived from LR, and multiclass classification has not been implemented for LR .

3.3.1.7 Naive Bayes (NB)

NB is available only for classification. To select NB, click the corresponding name. No hyperparameters need to be specified for this algorithm.

Note: in the current version of MVPANI, there will be no feature weights derived from NB.

3.3.2 Data Preprocessing: Feature Normalization

Sometime feature normalization is favorable before entering a machine learning model. Click the dropdown menu to select:

-None: Do not perform feature normalization.

-Normalization for Each Feature: Normalizing the values in each column (each column represents a feature) by transforming all values of each column to z scores with a mean of 0 and a standard deviation (SD) of 1 using the following equation:

$$z_{ij} = \frac{x_{ij} - \text{mean}(x_{.j})}{SD(x_{.j})} \text{ for the } j^{th} \text{ column.}$$

-Normalization for Each Sample: Normalizing the values in each row (each row represents a sample) by transforming all values of each sample to z scores with a mean of 0 and a standard deviation (SD) of 1 using the following equation:

$$z_{ij} = \frac{x_{ij} - \text{mean}(x_{i.})}{SD(x_{i.})} \text{ for the } i^{th} \text{ row.}$$

3.3.3 Data Preprocessing: Feature Dimensionality Reduction

Sometime feature dimensionality reduction is favorable for avoiding overfitting problem

before entering a machine learning model. Click the dropdown menu to select:

- None: Do not perform feature dimensionality reduction.

- PCA (Principal Components Analysis): When selecting the PCA method for dimensionality reduction, a window will pop up as shown in Figure 3.3.4-1, where you can specify the percentage of variance to be explained by the PCs you would like to keep. Here, a number between 0 and 100 should be entered; for example, 95 means that the kept PCs should explain at least 95% of the total variance of the original feature data, so that the number of PCs to be kept can be determined.

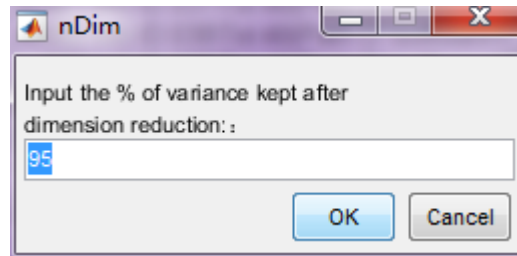


Figure 3.3.3-1 PCA parameter adjustment interface

3.3.4 Data Preprocessing: Feature Selection

Feature Selection has a similar role with Feature Dimensionality Reduction (Section 3.3.3) in avoiding overfitting problem. The difference between the two are: the actual features used in the machine learning model are not the original features any more but new features derived from the original features (e.g. PCs) after Feature Dimensionality Reduction, whereas the actual features used in the machine learning model are still the original features but only a subset of all original features after Feature Selection. Note that Dimensionality Reduction and Feature Selection cannot be used together; therefore, Feature Selection will be disabled if Feature Dimensionality Reduction is selected.

3.3.4.1 Methods for Feature Selection

Select the feature selection method from the dropdown menu:

- None: Do not perform feature selection.

- Lasso: Selecting features by LASSO (Least Absolute Shrinkage and Selection Operator) which uses the L_1 norm to make the model sparse.

- FScore: Selecting features according to the F-scores of features. The F-score of each feature is derived from an F test and measures the difference of the values of this given feature between the two classes based on the training samples, and thus is only available for classification tasks. The features with higher F scores indicate that these features are more discriminative between the

two classes in the training samples, and thus will be selected for training the classifier.

-Weight: Selecting features according to the weights of features. The weight of each feature is derived from a machine model trained using all features, and measures the contribution of this given feature to the trained model based on the training samples. The features with higher weights indicate that these features are more important for discriminating between the two classes in the training samples, and thus will be selected for training the final classifier. This option is only available for SVM/SVR with linear kernel and for binary classifications in the current version of MVPANI.

-Relieff: For classification tasks, the ReliefF algorithm is used; for regression tasks, the RReliefF algorithm is used. In Relief, a score is calculated for each feature which can be used to rank all features and then feature selection can be performed based on top scoring features. Relief feature scoring is based on the identification of feature value differences between nearest neighbor instance pairs. If a feature value difference is observed in a neighboring instance pair with the same class (a 'hit'), the feature score decreases; otherwise, if a feature value difference is observed in a neighboring instance pair with different classes (a 'miss'), the feature score increases.

For feature selection by LASSO, the number of selected features will be automatically determined. For feature selection by FScore, Weight, or Relieff, you can specify the number of selected features in two ways: (1) Specify the actual number of selected features directly, which requires an integer less than the total number of all features, e.g., 20 means selecting the top 20 features with the highest FScores/Weights/Relieff; (2) Specify a percentage, which requires a value between 0 and 1, e.g., 0.1 means selecting the top 10% of the features with the highest FScores/Weights/Relieff.

3.3.4.2 Specify a Series of Feature Numbers

When using feature selection by FScore, Weight, or Relieff, you may feel difficult to specify the number/percentage of selected features and would like to test a series of numbers/percentages of selected features to see how the MVPA results change with the number of features. This can be easily done in MVPANI by simply entering a series of numbers/percentages as a vector. The number of steps in a series of feature selection procedure is denoted by *m*. A certain step in this series will be called “a feature group” in this manual, and is denoted by *m*.

For example, if you enter "100:50:300", it means that the feature selection will start from 100 features up to 300 features with an increment of 50 features, i.e., five machine learning models will be trained and tested, each with a different number of selected features: 100, 150, 200, 250, 300. Similarly, if you enter "0.1:0.1:0.5", it means that the feature selection will start from 10% up to

50% of total features with an increment of 10%.

3.3.5 Searchlight MVPA

The Searchlight MVPA is designed for information localization and thus is only applicable for spatially structured data such as fMRI brain images or EEG time-frequency spectrum. Taking an fMRI brain image as an example, it is a voxel-wise analysis, that is, one MVPA is performed for every voxel within the whole brain (or a predefined brain region specified using the Mask file): when performing MVPA for a particular voxel, only the voxels (i.e., features) within a “searchlight” area which is often defined as a sphere or a cube centered at this given voxel, and such MVPA will run repeatedly for every voxel, resulting in a classification accuracy of the test dataset (or, for regression tasks, a correlation coefficient between the model-predicted values and the true values of the test samples) for each voxel. In this way, you will obtain a brain map of classification accuracies (or correlation coefficients) from which you can localize the areas containing useful information for your classification task (i.e., areas with high accuracies) or regression task (i.e., areas with high correlation coefficients).

Note that, only the voxels in the mask will be included in the searchlight area.

To perform Searchlight MVPA, check the corresponding box and specify the radius (denoted by r , in voxels) to define the size of the searchlight sphere (see Figure 3.3.5-1).

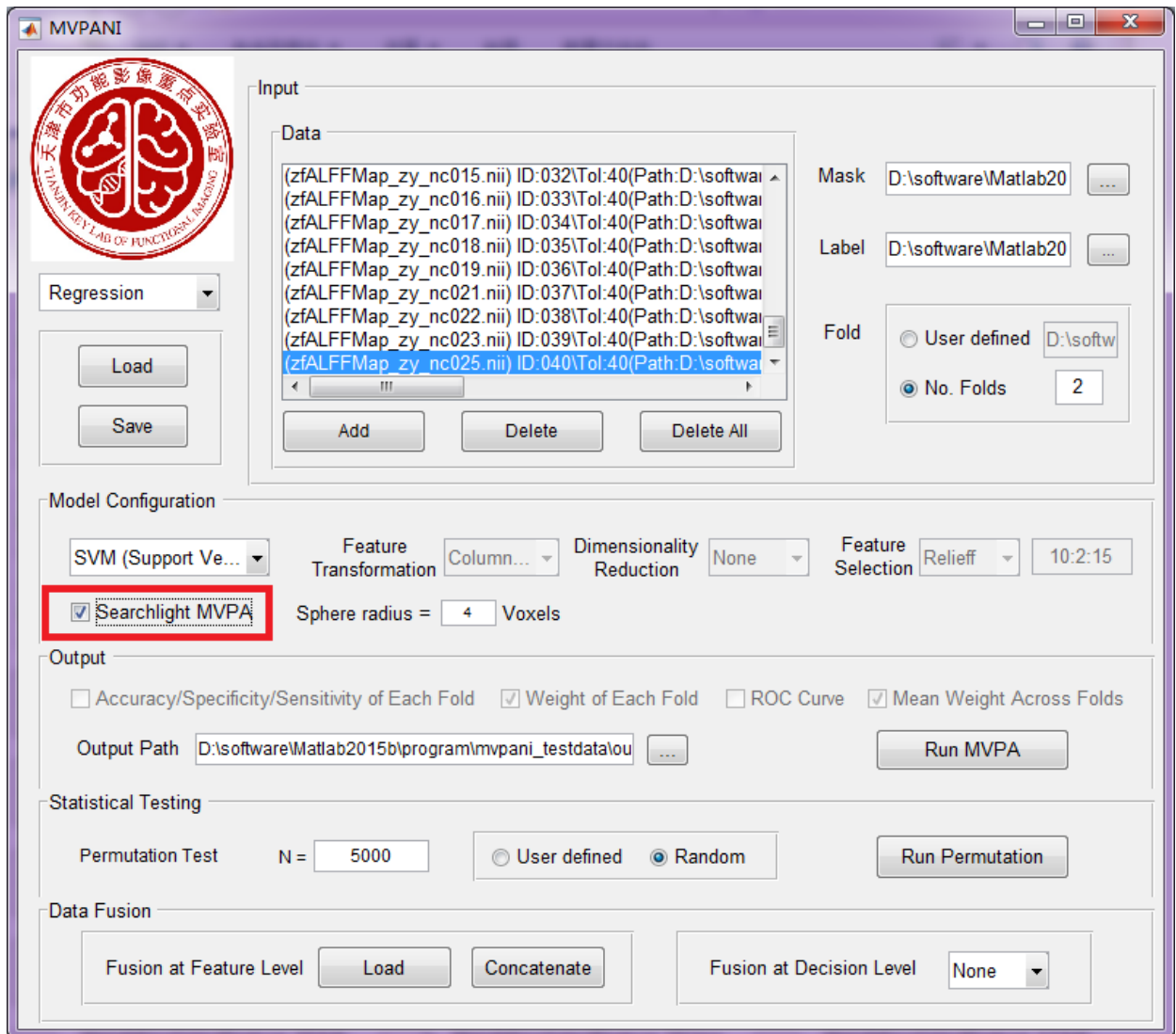


Figure 3.3.5-1 Example of configuration for Searchlight MVPA

In the current version of MVPANI, only the input data with NIfTI format (i.e., nii files) and the SVM method are allowed. Feature Normalization, Feature Dimensionality Reduction and Feature Selection will be disabled because these preprocessing steps are not needed in most cases.

Note that Searchlight MVPA often takes much longer time to finish as MVPA has to be run repeatedly for many times (e.g., the number of voxels within the whole brain).

3.4 Output Module

After you've finished data input and model configurations, you need to tell MVPANI where to save the output files by specifying an existing directory in the "output path" box and what outputs you would like to be saved in this output path (all output files will be saved in the working directory

by default if you leave the "output path" box blank). There are four optional outputs:

-Accuracy/Specificity/Sensitivity of Each Fold: This option is available only for classification tasks. When checked, the classification accuracy, specificity and sensitivity obtained for each fold during cross-validation will be saved in the variable named “SpecificitySensitivityAccEachFold” in MvpaResults.mat.

-Weight of Each Fold: This option is currently only available for SVM/SVR. When checked, the weight of each feature derived from the machine learning model trained during each cross-validation step will be saved. (Section 3.4.2.2)

-Mean Weight Across Folds: This option is currently only available for SVM/SVR. When checked, the mean weights averaged across all cross-validation steps will be saved. (Section 3.4.2.4)

-ROC Curve: This option is available only for classification tasks (all classification algorithms except Decision Tree). When checked, the ROC curve will be plotted and the corresponding figure will also be saved in the output path.

Note: ① The files in the output path folder will be automatically replaced with the newly generated output files with the same names. ② Do not close any result figure before all figures appear; otherwise, an error may occur when saving the result files.

Now that you have finished the data input, model configurations and output specifications, you can finally run the MVPA: just click the "Run MVPA" button; wait a few seconds when the MVPANI is loading in all your data, and then you will see real-time output information in the Command Window of Matlab and a popup window showing a progress bar (Figure 3.4-1).

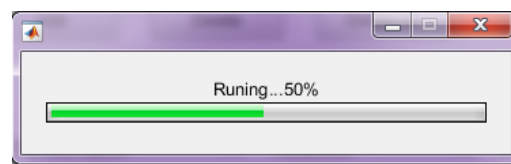


Figure 3.4-1 The popup window indicating the real-time progress of MVPA. It does not appear immediately after clicking the "Run MVPA" button, but only after all the data has been loaded.

3.4.1 The Main Outputs

After running the MVPA, two mandatory output files – MvpaResults.mat and RawData.mat – will appear in your output path folder.

3.4.1.1 MvpaResults.mat

MvpaResults.mat stores the information about your MVPA and the major results, including:

-FeatureNum: The number of features after Mask filtering.

-IndexColumn: Stores the indices of the features selected after Mask filtering in the original data files.

-Fold: A structure stores the fold information for cross-validation procedure. If the user has specified a Fold file, the file path and the "Fold" vector (i.e., the actual fold to which each sample belongs) are saved; if only the number of folds were specified, only the specified fold number is saved.

-MaskPath: Stores the path of the Mask file.

-Label: A structure stores the path of the Label file and the "Label" vector (i.e., the actual labels of all samples).

-SelectFeatureNum: The number of selected features after Feature Selection.

-Method: Indicates the type of the MVPA: classification or regression.

-SelectFeature: Indicates which Feature Selection method was used.

-FeatureTransformation: Indicates which Data Transformation method was used.

-ModelParameter: Stores the parameters used in the machine learning model.

-MeanAccuracy: The obtained mean classification accuracy averaged across all folds (i.e., all cross-validation steps). For example, MeanAccuracy = 75.3 means the mean accuracy is 75.3%. Only available for classification.

-Predict_test_Label: A $n \times a$ matrix, and a varies with the number of categories and the machine learning algorithm. Each row represents a sample. 1st column is true labels, 2nd column is predicted labels. For SVM, the number of columns $a = (2 + C_l^2)$, and C_l^2 is the combination symbol. For example, $C_5^2 = \frac{5!}{2! \times (5-2)!} = 10$. The 3rd column to the last column store the decision value of the binary classification SVM model corresponding to each combination. For other machine learning algorithms other than SVM, $a = l$. The 3rd column to last column store the probabilities of each class obtained from the machine learning model. Only available for classification.

-ClassAlgorithm: The classification algorithm used

-DimensionalityReduction: Stores the information about the dimensionality reduction method.

-AUC: The obtained AUC value if the option "ROC Curve" was checked, otherwise the value is 0. Only available for binary classification (i.e., $l = 2$).

-SpecificitySensitivityAccEachFold: A $k \times 3$ matrix stores the specificity (1st column), sensitivity (2st column), and accuracy (3st column) of each fold if the option

"Accuracy/Specificity/Sensitivity of Each Fold" was checked. Only available for classification and $l = 2$.

-MeanSpecificitySensitivityAcc: A vector stores the specificity (1st), sensitivity(2st) and accuracy(3st) averaged across all folds (i.e., averaged over all cross-validation steps). Only available for classification and $l = 2$.

-AccuracyEachClassEachFold: A $k \times (l + 1)$ matrix – the 1st to the l st columns store the accuracies of each class in each fold, and the last column stores the average accuracy of each fold across all classes. Only available for multi-class classifications ($l > 2$).

-MeanAccuracyEachClassEachFold: A vector stores the mean accuracy of each class across all folds (the first l columns) and the average accuracy across all classes and all folds (the last column). Only available for multi-class classifications ($l > 2$).

-confusion_matrix: A $l \times l$ matrix stores the confusion matrix across all folds (i.e., obtained from all samples). The rows represent the true labels and the columns represent the predicted labels. The confusion matrix will also be saved as bmp format for visualization. Only available for multi-class classifications ($l > 2$).

-GroupFile: A cell stores the path information of all sample data files.

-r: Pearson's correlation coefficient between the predicted values and the true values of all samples by pooling the test samples of all folds together. Only available for regression.

-p: The P-value (i.e., statistical significance) of the obtained Pearson's correlation coefficient between the predicted values and the true values of all samples. Only available for regression.

-predLabel: A $n \times 2$ matrix stores the true values (1st column) and the predicted values (2nd column). Each row represents a sample. Only available for regression.

If you specified a series of feature numbers (see Section 3.3.4.2), an extra dimension with size m will be added to some specific variables (SelectFeatureNum, AUC, MeanAccuracy, SpecificitySensitivity, Predict_test_Label, SpecificitySensitivityAccEachFold, MeanSpecificitySensitivityAcc, AccuracyEachClass, AccuracyEachClassEachFold, MeanAccuracyEachClassEachFold, confusion_matrix, r, p, predLabel) to store the results obtained from different selected feature groups.

3.4.1.2 RawData

RawData.mat stores a matrix corresponding to the unprocessed sample data but filtered by the Mask and the data of each sample was flattened into a 1-D row vector. Each row represents a sample and each column represents a feature.

3.4.1.3 Accuracy/Correlation Coefficient Line Graph

If you specified a series of feature numbers (see Section 3.3.4.2), the classification accuracies (or the correlation coefficients between the predicted values and the true values if it's a regression task) obtained based on all specified feature numbers will be output as a line graph (see Figure 3.4.1.3-1 for a classification task or Figure 3.4.1.3-2 for a regression task). The corresponding figure will also be saved in the Output Path in two file formats (bmp and fig).

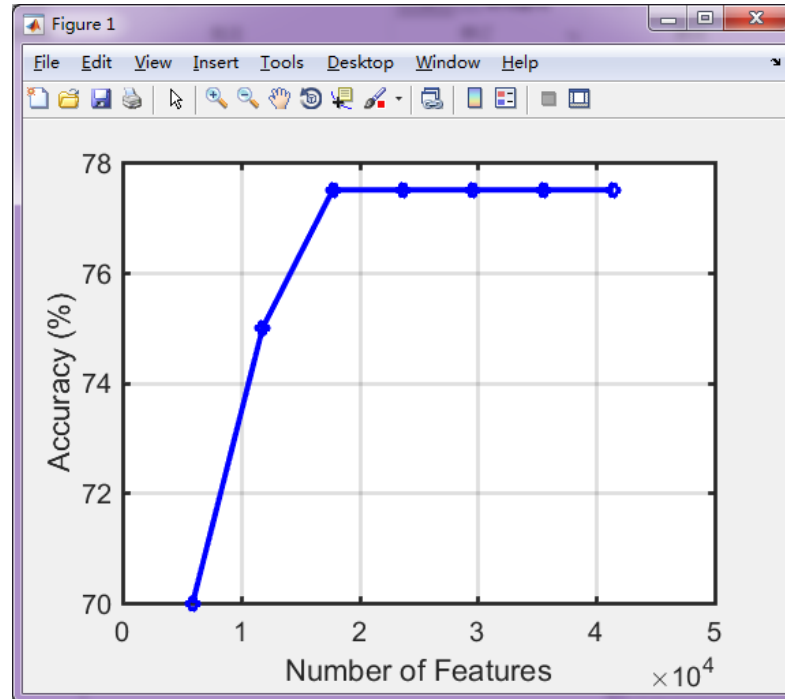


Figure 3.4.1.3-1. The output classification accuracy line graph for classification tasks.

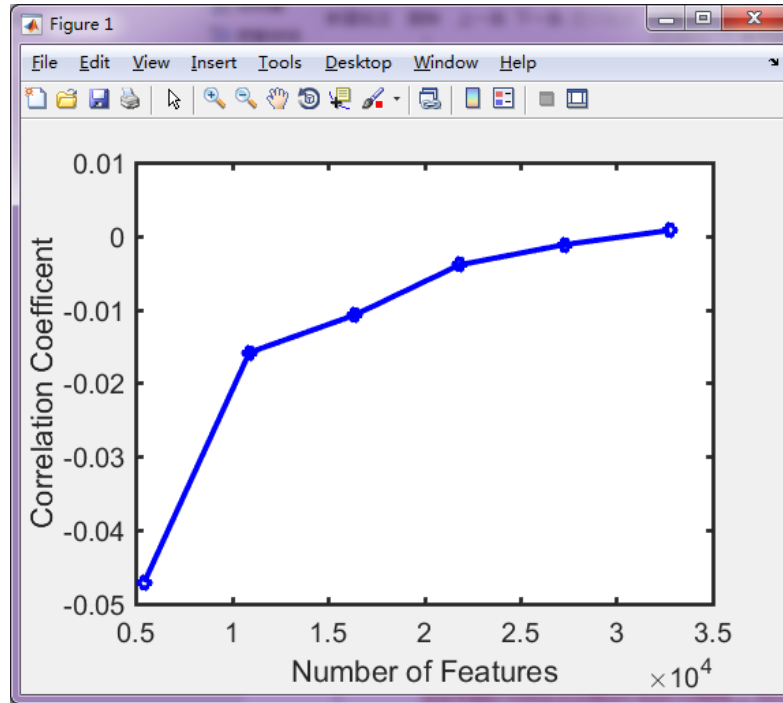


Figure 3.4.1.3-2. The output correlation coefficient line graph for regression tasks.

3.4.1.4 Confusion Matrix

If a multi-class classification is performed, the resultant confusion matrix will be saved in MvpaResults.mat (Section 3.4.1.1) and as an image (i.e., .bmp and .fig format) for viewing. The output files include:

ConfusionMatrix_*m*.bmp/fig: Confusion matrix saved in image format (Figure 3.4.1.4-1).

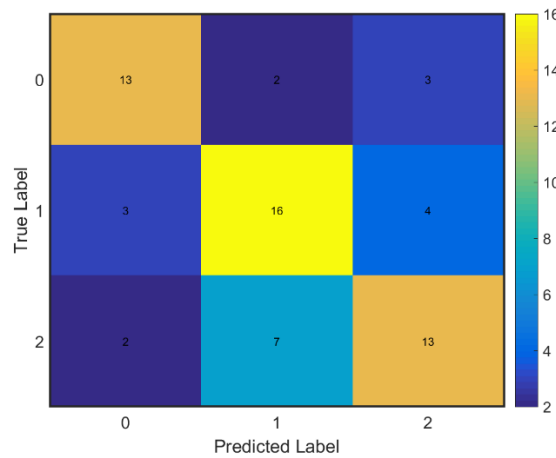


Figure 3.4.1.4-1. The figure of confusion matrix. Rows are true labels and columns are predicted labels.

3.4.2 Other Optional Outputs

3.4.2.1 Accuracy/Specificity/Sensitivity of Each Fold

This optional output is available only for classification tasks.

If the output option “Accuracy/Specificity/Sensitivity of Each Fold” was checked, the variable "SpecificitySensitivityAccEachFold" will be added to the main results file MypaResults.mat, which stores the sensitivity, specificity, and accuracy of each fold of the cross-validation procedure.

3.4.2.2 Feature Weights of Each Fold

If the option “Weight of Each Fold” is checked, the following outputs will be obtained (Figure 3.4.2.2-1):

-SelectFeature_ \hat{m} _Fold_ \hat{k} .mat: This file will be saved when the option "Feature Selection" is specified and the sample data files are not in the NIfTI format. In the actual filename, \hat{m} is replaced with a certain step in a series of feature selection procedure, and \hat{k} is replaced with a certain step during cross-validation. This file contains a binary matrix with the same shape as the raw sample data input. 1 means that the corresponding feature is selected and 0 means that the feature is not selected. The content of this file is equivalent to FeatureAll.mat but with the shape of the raw sample data.

-SelectFeatureImg_ \hat{m} _Fold_ \hat{k} .mat/.nii: When the option "Feature Selection" is specified and the input data is in NIfTI format, these two files (one in mat format and the other in nii format) will be saved instead of SelectFeature_ \hat{m} _Fold_ \hat{k} .mat – adding "Img" in the filename to indicate the input data are images.

-WeightSelectFeature_ \hat{m} _Fold_ \hat{k} .mat: This file will be saved when the option "Feature Selection" is specified and the sample data files are not in the NIfTI format and only available for SVM. It contains a matrix of the same shape as the input raw sample data and stores the feature weights obtained in the \hat{m} th feature selection group and the \hat{k} th fold of the k -fold cross-validation procedure.

-WeightImgSelectFeature_ \hat{m} _Fold_ \hat{k} .nii: Only generated when the input data is in NIFTI format. These files are equivalent with WeightSelectFeature_ \hat{m} _Fold_ \hat{k} .mat but in NIFTI format.

When “Feature Selection” is not specified, "SelectFeature_ \hat{m} _" in the name of the above files

will be replaced by "Feature".

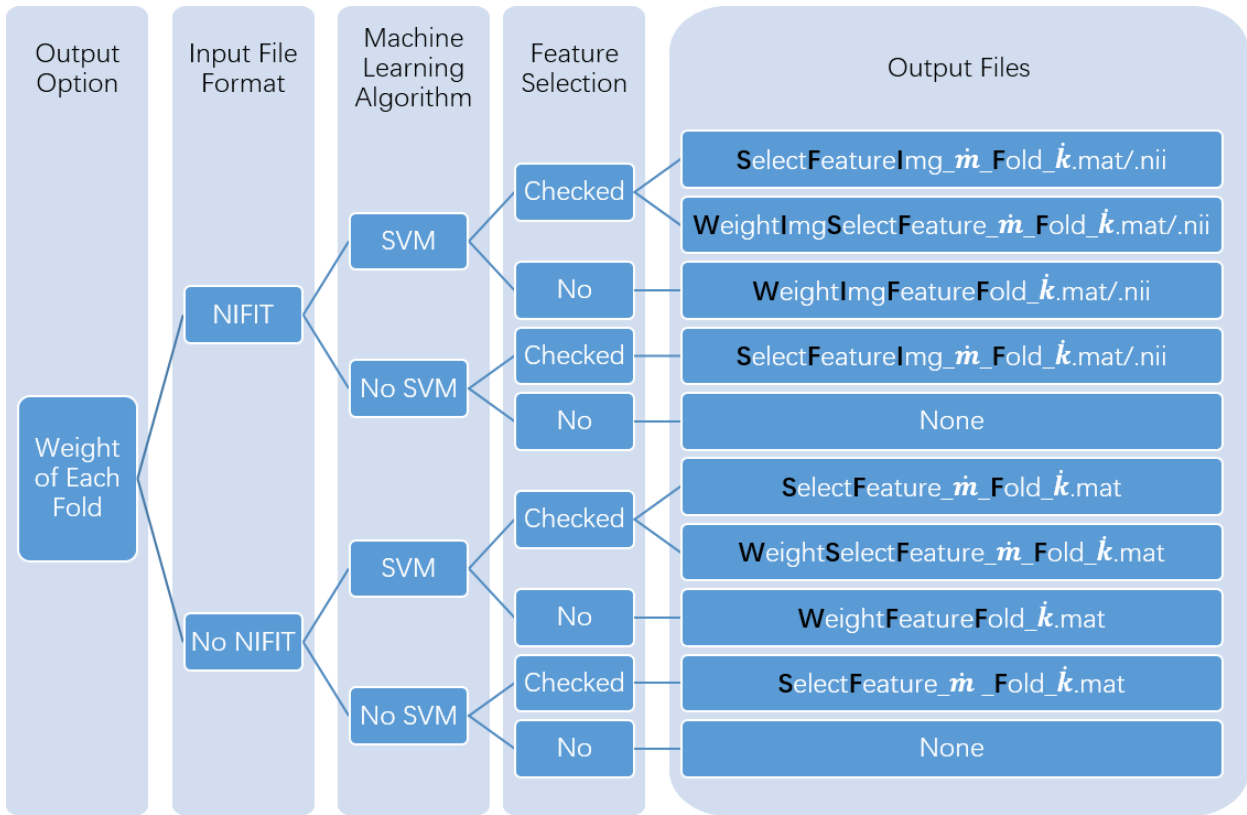


Figure 3.4.2.2-1 The output files generated in different scenarios when “Weight of Each Fold” is checked..

3.4.2.3 ROC Curve

This optional output is only available for binary classification tasks. When this option is checked, the ROC curve will be output as a figure (see Figure 3.4.2.3-1) and saved in both bmp and fig formats (ROC.bmp and ROC.fig). If a series of feature numbers are specified, the ROC curve obtained from every specified feature number will be plotted within one figure (see Figure 3.4.2.3-2).

ROC.bmp: ROC curve saved in image format.

ROC.fig: ROC curve image saved in MATLAB format.

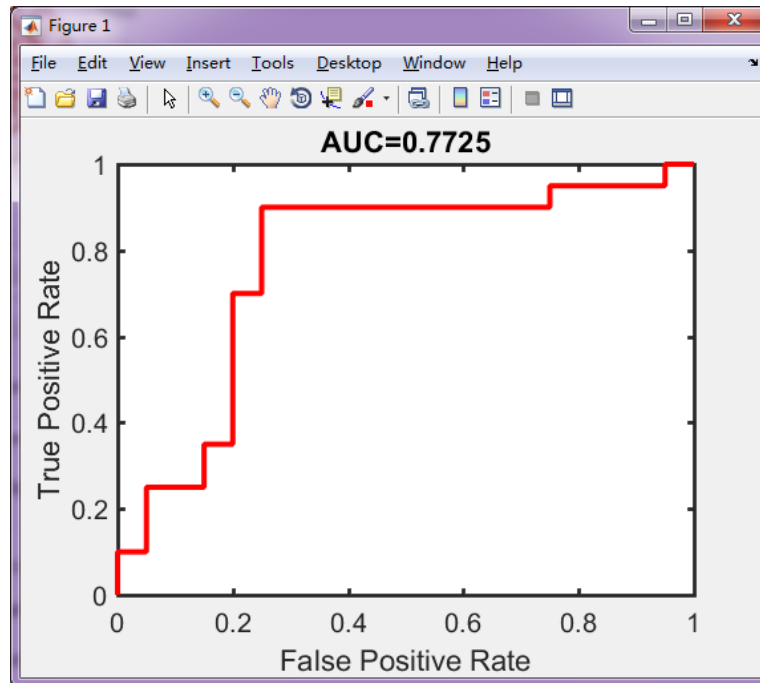


Figure 3.4.2.3-1 The ROC curve.

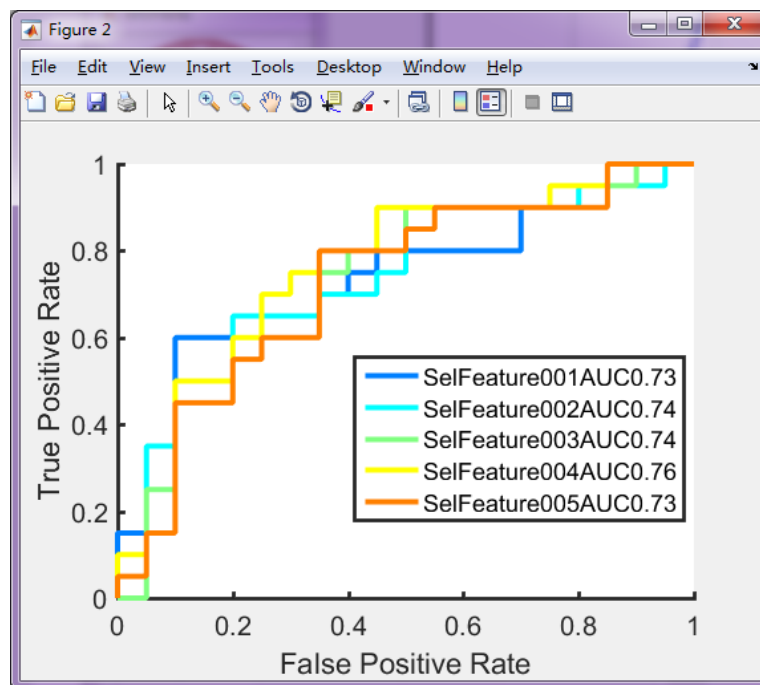


Figure 3.4.2.3-2 The ROC curves obtained from every specified feature number.

3.4.2.4 Mean Weights Across Folds

When the option “Mean Weight Across Folds” is checked, the average weights and average feature selection results of the cross-validation will be generated, which correspond to the mean

values of the relevant outputs in "Feature Weights of Each Fold " (Section 3.4.2.2). According to your configuration, the following outputs will be obtained (Figure 3.4.2.4-1):

(1) Percentage of features selected in cross-validation

-SelectFeaturePercentage_*m*.mat (or SelectFeaturePercentageImg_*m*.nii/mat): This file will be saved when the option "Feature Selection" is specified. This file contains a matrix with the same shape as the raw sample data input, and the value of each entry represents the percentage (i.e., how many times) that the corresponding feature (voxel) was selected in the whole cross-validation procedure. For example, if a feature was selected three times in ten-fold cross-validation, the value of this feature in this file will be 0.3. The *m* indicates that it is the result of the *m*th feature group. If the input files are in NIfTI format, the name of this output file will be "SelectFeaturePercentageImg_*m*.nii/mat".

(2) Average weight of each feature selection group

-WeightFeature: Storing the average weight of each feature across all cross-validation steps. If the input file is in NIfTI format and the option "Feature Selection" is checked, the name of the output files will be "WeightImgSelectFeature_*m*.nii/mat". If the input files are in NIfTI format and the option "Feature Selection" is not checked, the name of the output files will be "WeightImgFeature.nii/mat". If the input files are not in NIfTI format and the option "Feature Selection" is checked, the name of the output files will be "WeightSelectFeature_*m*.mat". If the input file is not in NIfTI format and the option "Feature Selection" is not checked, the name of the output files will be "WeightFeature.mat". (Figure 3.4.2.4-1).

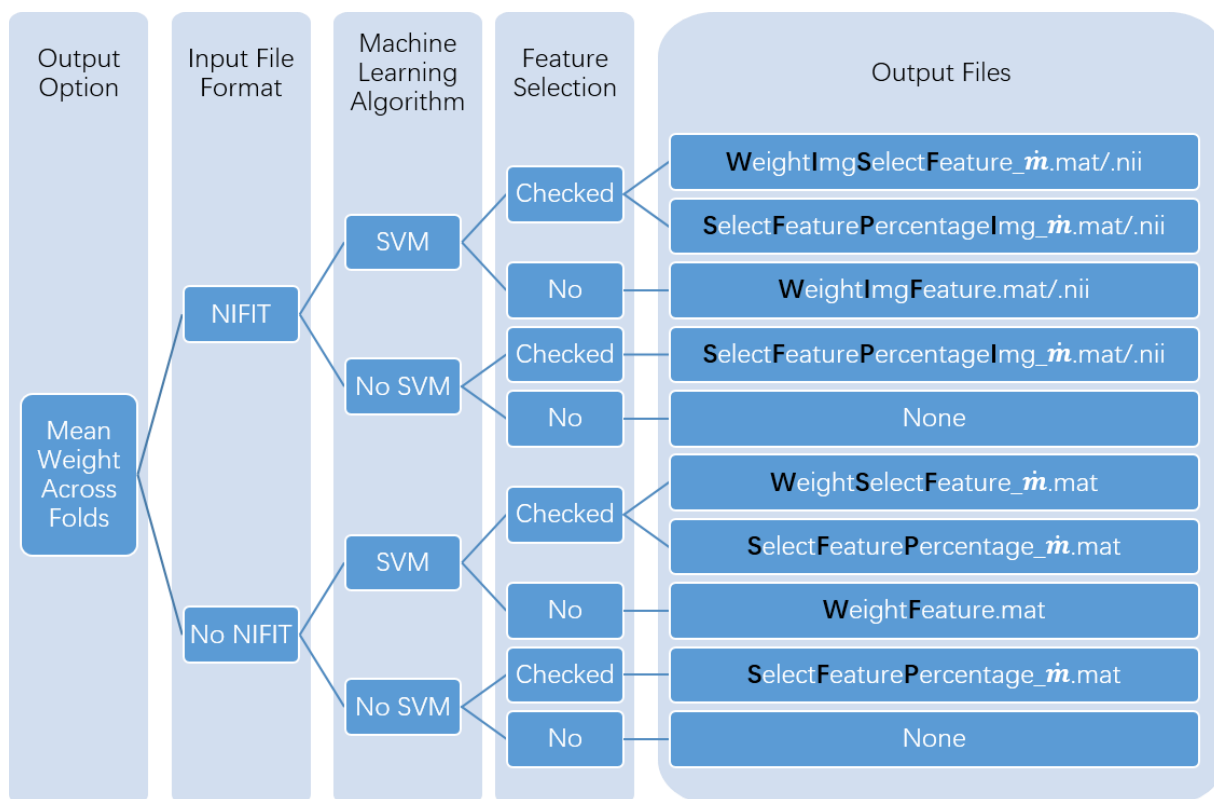


Figure 3.4.2.4-1 The output files generated in different scenarios when “Mean Weight Across Folds” is checked.

3.4.3 Outputs for Searchlight MVPA

After running searchlight MVPA, a file named “RawData.mat” and a folder named “results” will appear in your output folder.

The RawData.mat is the same as in Section 3.4.1.2, storing a matrix corresponding to the unprocessed sample data but filtered by the Mask.

The folder “searchlight” in the folder “results” stores the results of searchlight MVPA, and contains the following files. The file “res_accuracy_minus_chance.hdr/.img” (Classification) or “res_corr.hdr/.img” (Regression) stores the main results and the remaining files stores relevant information about the configuration of the performed Searchlight MVPA.

(1) cfg.mat: A structure stores the configuration of searchlight MVPA, output by the toolbox “decoding_toolbox_v3”.

(2) res_accuracy_minus_chance.hdr/.img: The main output of searchlight MVPA. For classification, it stores the subtraction of average classification accuracy minus the random accuracy (i.e., 0.5) in each voxel (Figure 3.4.1-1). Only available for classification.

(3) res_corr.hdr/.img: Same as res_accuracy_minus_chance.hdr/.img, but stores the correlation coefficient between the predicted values and the true values in each voxel. Only available for regression.

(4) res_accuracy_minus_chance/res_corr.mat: A structure stores the result of searchlight MVPA, output by the toolbox “decoding_toolbox_v3”.

(5) res_cfg.mat: A structure similar to cfg.mat but stores a few extra information such as the size of the image and the size of voxels, output by the toolbox “decoding_toolbox_v3”.

(6) res_filedetails.txt: This file stores the fold information for cross-validation procedure.

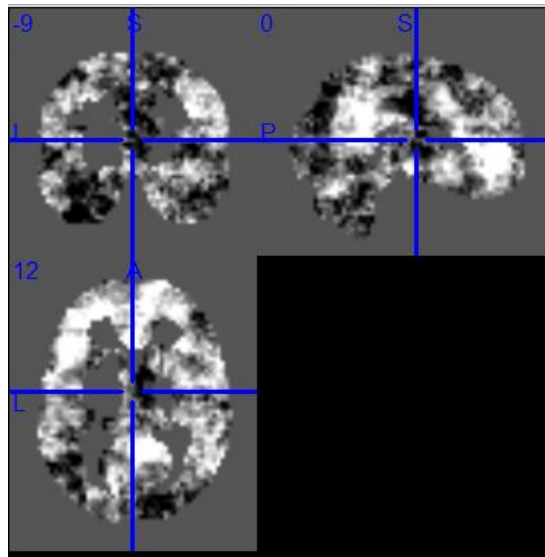


Figure 3.4.2.4-1 The resultant brain map of classification accuracies obtained from Searchlight MVPA, stored in the file “res_accuracy_minus_chance.hdr/.img”.

Note: All optional outputs such as “Accuracy/Specificity/Sensitivity of Each Fold”, “Feature Weights of Each Fold”, “ROC Curve”, “Mean Weights Across Folds” are not available for Searchlight MVPA.

3.5 Statistical Testing Module

Permutation test is implemented in MVPANI for testing the statistical significance of the obtained classification accuracy (for classification tasks) or correlation coefficient (for regression tasks); therefore, you must first run a MVPA and the output file, MvpaResults.mat, has been generated in the specified Output Path.

Note: please make sure that all configurations in the Model Configuration Module for permutation test must be identical with the configurations you used when the corresponding

MvpaResults.mat file was generated. This can be easily ensured in two ways: (1) Run permutation test immediately after the main MVPA has finished (and before you make any changes to the Model Configuration); (2) Use the Save/Load function (see Section 3.1.2) to reload your previously saved model configuration (in case you did not save your model configuration information, you can still find all required information in the MvpaResults.mat).

3.5.1 Permutation Test Configuration

You need to tell MVPANI three things in order to run a permutation test:

-Path to the corresponding MvpaResults.mat file: The file path specified in the Output Path box in the Output Module must contain the MvpaResults.mat file. The results generated from the permutation test will also be saved in the same path.

Specifically, for searchlight MVPA permutation test, the file path specified in the Output Path box must contain the “results” folder, which is the output folder of searchlight MVPA.

-Number of permutations (N): Enter an integer in the box beside the text “N =” to indicate how many permutations you want to run for this permutation test. Note that, the larger the N , the more precise the generated null distribution; therefore, a larger N is recommended (usually at least a few thousands). The default value is 5000. However, larger N will also result in longer computation time; therefore, you also need to consider your hardware configuration when you decide the value of N . We will use \hat{N} to denote the \hat{N} th permutation in a permutation test.

-Randomly permuted labels: There are two ways to specify the random labels of the training samples after permutation (note that, in each cross-validation step, only the labels of training samples will be randomly permuted):

- “User defined”: you can specify a pre-defined random label file (after random permutation) by clicking the “User defined” option. The supported file formats include xlsx, xls, mat, and txt. The file content should be a matrix with a size of $n \times N$, where n is the number of samples and N is the number of permutations; that is, each row represents a sample and each column represents a permutation.
- “Random”: you can simply ask MVPANI to automatically perform a random permutation on the labels of training samples for each permutation and cross-validation step (default option).

If you want to repeat a permutation test using exactly the same randomly permuted labels used in a previous permutation test but you do not want to generate your own randomly permuted label file, you can use the “Random” mode at the first time and the randomly permuted labels generated by MVPANI will be saved as a file “RandLabel.mat” in the Output Path, and then you

can use the "User defined" mode to load this file and run your second permutation test.

Once these configurations are done, just click the button "Run Permutation" to start the permutation test. A dialog box will pop up and you need to enter the number of CPU cores you would like to utilize for parallel processing so that you can accelerate the permutation test (the actual number of cores needs to be decided based on your hardware configuration – how many cores on your computer are available for running the permutation test).

In particular, for the searchlight MVPA permutation test, after the permutation test finishes, an extra dialog (See Figure 3.5.1-1) will pop up, where you can enter a number p ($0 < p < 1$), which is the P value threshold for determining the statistical significance of the searchlight MPVA results. Note that this threshold corresponds to the threshold after correction for multiple comparisons (i.e., the number of voxels in the mask).

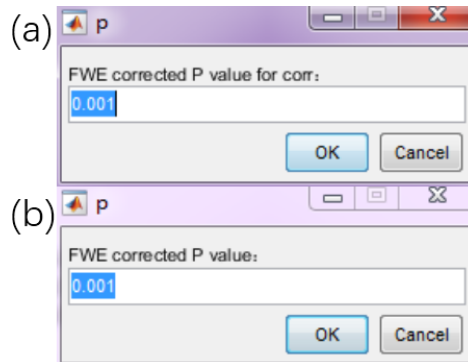


Figure 3.5.1-1 Enter the threshold of statistical significance for the permutation test in searchlight MVPA: (a) for regression, (b) for classification.

3.5.2 Outputs of Permutation Test

When the permutation test is completed, the following outputs will be generated:

(1) `PermutationClassAccuracy_feature_m.bmp/fig`: The histogram of the chance-level classification accuracies (for classification tasks) or correlation coefficients (for regression tasks) generated from the N permutations will be plotted (see Figure 3.5.2-1) and saved in the Output Path. The actual classification accuracy/correlation coefficient obtained from the true labels is also shown as a red vertical line in this plot. The resultant P value will also be given at the top of the figure – it is calculated as the percentage of the chance-level classification accuracies/correlation coefficients that are equal to or greater than the actual classification accuracy/correlation coefficient (if none out of N permutations reached the actual classification accuracy/correlation coefficient, the P value is denoted as $P < 1/N$ instead of $P = 0$ because the precision of the calculated P values is dependent on the number of permutations).

(2) RandLabel.mat: This file will be saved and contains all randomly permuted labels generated by MVPANI if the "Random" option is selected. It is a matrix with a size of $n \times N$, where n is the number of samples and N is the number of permutations; that is, each row represents a sample and each column represents a permutation.

(3) MvpaPermResults: This file contains a structure storing all MVPA results obtained from the permutation test:

-PermutationMeanClass: A matrix stores the chance-level mean classification accuracy/correlation coefficient (averaged across all folds) obtained from each permutation. The size of the matrix is $N \times m$, where N is the number of permutations and m is the number of feature selections (see Section 3.3.4.2); that is, each row corresponds to a permutation and each column corresponds to a feature selection. If you are not satisfied with the plot of the histogram generated by MVPANI, you can plot your own histogram using the values stored in this variable.

-DecValues: A matrix stores all predicted class labels (for classification tasks) or values (for regression tasks) obtained from each permutation. The size of the matrix is $m \times n \times N$, where m is the number of feature selections, n is the number of samples, and N is the number of permutations.

-IndexColumn: A vector stores the indices of the features selected after Mask filtering in the original data files, same as in MvpaResults.mat (see Section 3.4.1.1).

-Permutation_p: The P-value obtained by comparing the actual classification accuracy/correlation coefficient with its null distribution generated by the permutation test. It is calculated as the percentage of the chance-level classification accuracies/correlation coefficients that are equal to or greater than the actual classification accuracy/correlation coefficient (if none out of N permutations reached the actual classification accuracy/correlation coefficient, the P value is denoted as $P < 1/N$ instead of $P = 0$ because the precision of the calculated P values is dependent on the number of permutations).

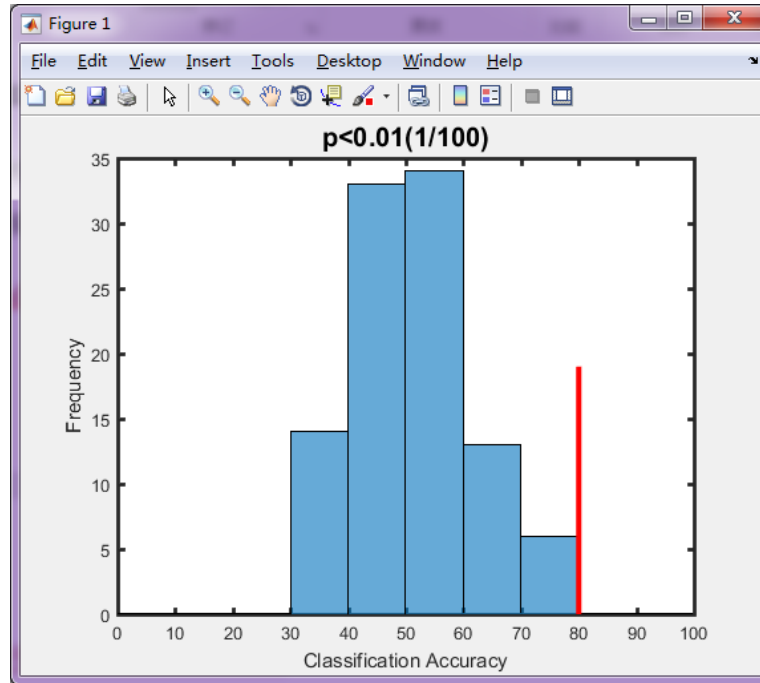


Figure 3.5.2-1. The histogram of the chance-level classification accuracies obtained from permutation test (100 permutations). The x-axis is the classification accuracies. The y-axis is the frequency. The red vertical line indicates the actual classification accuracy obtained using the true labels. The P-value is provided at the top: "p<0.01(1/100)" means that all chance-level accuracies obtained from the 100 permutations are lower than the actual classification accuracy.

3.5.3 Outputs of Permutation Test for Searchlight MVPA

The same interface described in Section 3.5.1 is used for permutation test for Searchlight MVPA. Before running permutation test for Searchlight MVPA, a folder named "results" contains a folder named "searchlight" (see Section 3.4.1) is required in the specified "Output Path".

When the permutation test for searchlight MVPA is completed, the following outputs will be generated in the output path:

(1) RandLabel.mat: Same as in Section 3.5.2. (2) a folder named "permutation": containing the main results of permutation test, and will be created in the existing "results" folder. It contains the following:

- N subfolders named "permutation_ N ": The content of Each subfolder contains the output files generated in the corresponding permutation step (please refer to Section 3.4.1 for the output files).

- permutation_FWE_Pvalues.mat/.nii: The mat file contains a vector storing the FWE corrected P-value of each feature (i.e., voxel), and the nii file contains the same P values as a brain

image.

The P-values are calculated as follows. ① For each permutation, the highest classification accuracy among all voxels is taken, resulting in a total of N classification accuracies when N permutations are completed. ② The null distribution is constructed using these N classification accuracies. ③ The corrected P-value of each voxel will be calculated as the percentage of the chance-level classification accuracies that are equal to or greater than the actual classification accuracy of this voxel.

-permutation_FWE_corrected_P_0.05_accuracy.nii: A brain map stores the voxels surviving the threshold of FWE corrected $P < 0.05$. The value of each voxel represents the classification accuracy (averaged across all folds) obtained from the Searchlight sphere centered at this voxel. Only available for classification.

-permutation_FWE_corrected_P_ p _accuracy.nii: Similar to the previous file but the statistical threshold p can be set manually. Only available for classification.

-permutation_corr_FWE_corrected_P_ p _coefficient.nii: A brain map stores the voxels surviving the threshold of FWE corrected $P < p$. The value of each voxel represents the correlation coefficient between the true values and the predicted values (averaged across all folds) obtained from the Searchlight sphere centered at this voxel. Only available for regression.

Note that, in these files, the value 0 represents a $P < 1/N$, the value -1 means that this voxel was not included in the mask, and the value -2 means that this voxel did not survive the threshold p .

3.6 Data Fusion Module

When different types of data are available, MVPA can fuse different types of data containing complementary information to improve classification or regression performance. MVPANI offers two strategies for data fusion: "Fusion at Feature Level" and "Fusion at Decision Level".

3.6.1 Fusion at Feature Level

In this strategy, different types of data are fused at the feature level before MVPA by first building a fused feature vector by concatenating the feature vectors obtained from different data types and then feeding the fused feature vector to the machine learning model. Therefore, the MVPA is only performed once using the concatenated features. Note that, as different types of data often have very different ranges, it is recommended that the data of each type should be normalized

using, for example, Z-transformation, to a similar range before concatenating them into a single feature vector. This can be done through the following steps:

3.6.1.1 Loading Data

In the sub-module of "Fusion at Feature Level", click on the button 'Load' and a dialog box pops up where you can add all sample data files of the first data type (Figure 3.6.1.1-1). Again, you can pre-select a subset of features by specify a Mask file. Click the "Done" button once you've added all data files of the first type.

Repeat the above procedure to add all data files of the second data type.

Repeat the above procedure for each data type until the data files of all types are loaded.

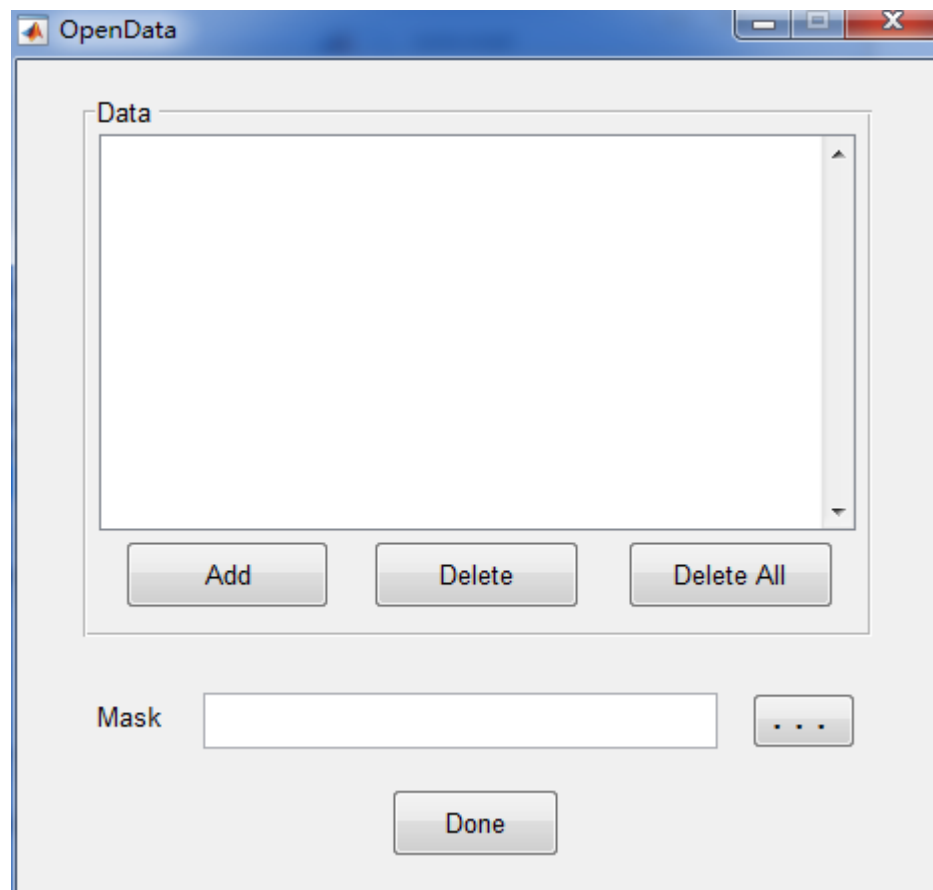


Figure 3.6.1.1-1 The dialog box for adding the data files for one type.

Note that during this process, make sure that the order of the added files of each data type must be the same for all data types; for example, the first file added for each data type must correspond to the same sample. If you made a mistake before the files of all data types have been loaded, you need to exit MVPANI and restart the whole procedure.

3.6.1.2 Generating the Fused Feature Vector

Once the files of all data types have been loaded, click the "Concatenate" button to generate the sample data files of the fused feature vector which will be saved in the folder of the Output Path (the fused feature data of each sample will be saved as a separate mat file: FeatureFusion_*n*.mat).

An additional file, FeatureFusionFile.txt, storing the path information of all added files, will also be saved in the Output Path.

3.6.1.3 Run MVPA Using Fused Feature Data

Once the fused sample data files have been generated, you can add these newly generated sample data files in the Input Module, and start a regular MVPA procedure described in Sections 3.2-3.5.

3.6.2 Fusion at Decision Level

In this strategy, individual machine learning models are built for each type of data separately, resulting in multiple classifiers or regression models. Each classifier or regression model makes its own decision for each test sample. Then the decisions from all classifiers or regression models will be fused to make the final decision. This can be done through the following steps:

Note that the subject number and the label must be the same for each modality.

3.6.2.1 Run MVPA for Each Type of Data

You need to obtain the decisions for all test samples made using each type of data before you can fuse these decisions. Just run a regular MVPA described in Sections 3.2-3.4 using each type of data and you will obtain a MvpaResults.mat for each data type – make sure that you rename this file or output the result file of different data type to different folder to avoid overwriting the previous result file by the subsequent one. Again, make sure that the order of the samples specified during the MVPA of each data type must be the same for all data types.

3.6.2.2 Fusing the Decisions Obtained From All Data Types

Click the dropdown menu in the sub-module of "Fusion at Decision level" to select the strategy for decision fusion, and then a window will pop up where you need to select the MVPA

result files obtained from all data types (e.g., MvpaResults_01.mat, MvpaResults_02.mat, MvpaResults_03.mat, etc.) (Figure 3.6.2.2-1).

For classification, MVPANI offers two strategies for decision fusion:

- Unweighted Vote: This option is the classical voting strategy by taking the majority of the predicted class labels from all classifiers as the final predicted class label. For example, if you have three classifiers (each trained using a different data type) and the predicted class labels of two classifiers are Class A and the predicted class label of the other classifier is Class B, then the final decision will be Class A.

- Weighted Vote: This option is a modified voting strategy by weighting the decision of each classifier during “voting” – the weight corresponds to how confident the classifier made its decision about a given sample and is usually the probability value generated by the machine learning algorithm (for SVM, it is the distance between the given sample and the decision hyperplane).

For regression, MVPANI simply average the predicted values obtained from all data types to generate the final predicted value.

If you specified a series of Feature Selections (Section 3.3.4.2) when you run the regular MVPA for each type of data, the resultant MvpaResults.mat files can also be fused as long as the specifications of Feature Selection series are the same.

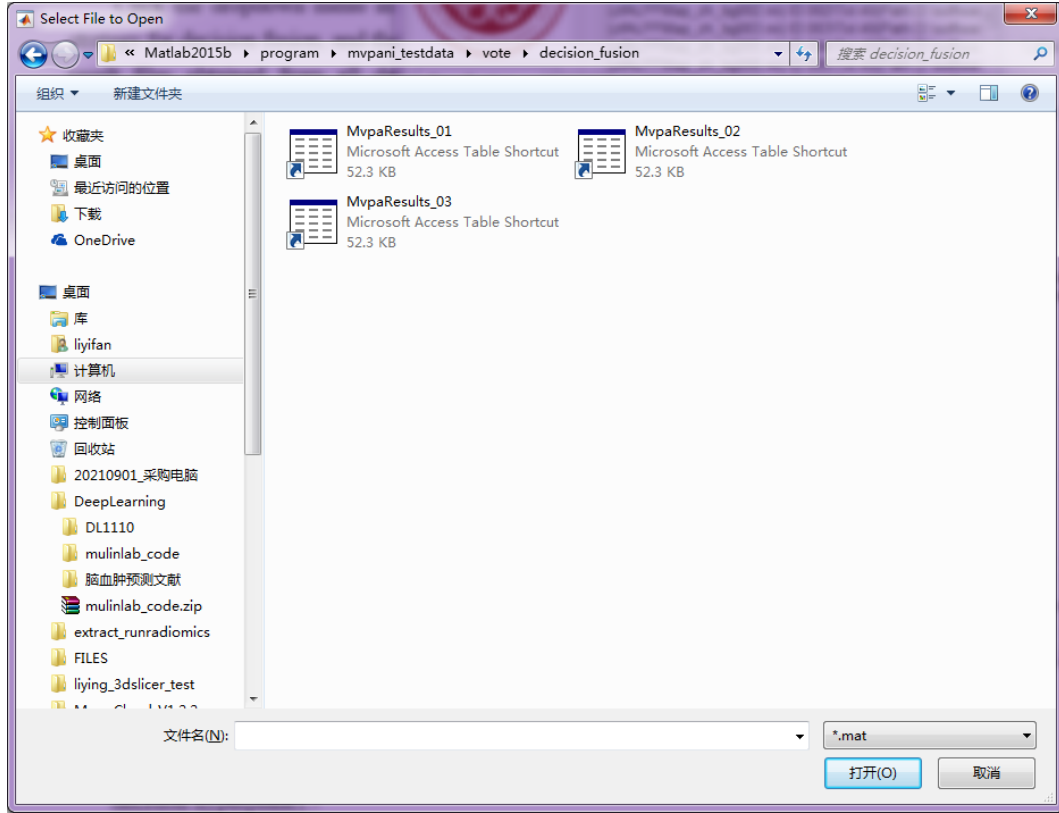


Figure 3.6.2.2-1 The dialog window for selecting multiple MVPA Results file for Decision Fusion.

3.6.2.3 Outputs of Unweighted Vote

“Unweighted Vote” can only be used for classifications. When “Unweighted Vote” is selected, the following outputs will be generated and saved in the Output Path.

(1) UnWeightedVoteResults.mat is a structure storing the main results and some relevant information, including:

- confusion_matrix (only available for multi-class classifications): the resultant confusion matrix (size: $l \times l \times m$).

- specificity_sensitivity (only available for binary classifications): A matrix (size: $m \times 2$) stores the final specificity (the 1st column) and sensitivity (the 2nd column) after fusion. Each row corresponds to a given Feature Selection.

- pred_label_vote: A matrix with the size of $n \times m$ (n is the number of samples and m is the number of Feature Selections) stores the final decisions of all samples; each column contains the final decisions of all samples after fusion for a given Feature Selection.

- pred_label: A matrix with the size of $n \times m \times f$ (n is the number of samples, m is the number of Feature Selections, and f is the number of separate models to be fused) stores the decisions of

all separate models.

-true_label: A column vector (size: $1 \times m$) stores the true labels of all samples.

-Accuracy_vote: A row vector (size: $1 \times m$) stores the final classification accuracy after decision fusion for a given Feature Selection.

Note: If “Feature Selections” is not specified, the above dimensions related to m will be removed or set to 1.

(2) ConfusionMatrix_ m .bmp/.fig is the confusion matrix of fusion results (Figure 3.6.2.3-1), only available for multi-class classifications.

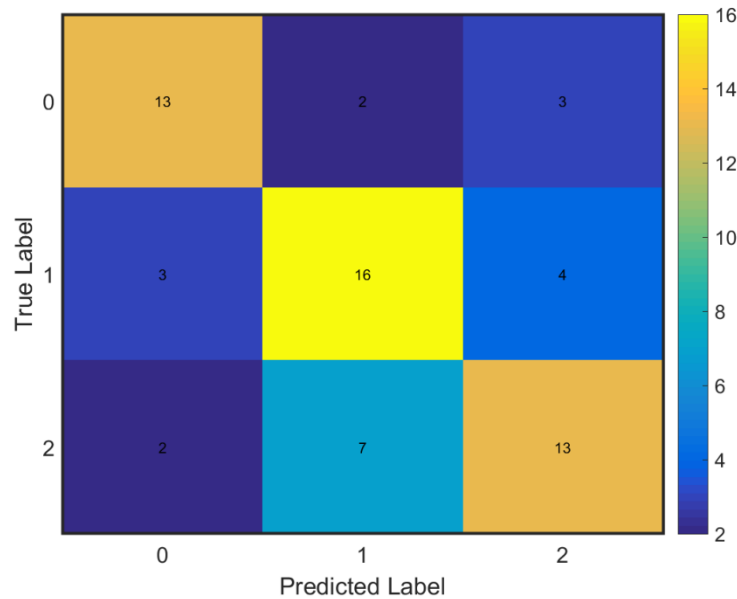


Figure 3.6.2.3-1 Confusion matrix of fusion results in multi-classification

3.6.2.4 Outputs of Weighted Vote

“Weighted Vote” can be used for both classifications and regressions. Then “Unweighted Vote” is selected, the following outputs will be generated and saved in the Output Path.

(1) WeightedVoteResults.mat is a structure storing the major results and relevant information, including:

-confusion_matrix (only available for multi-class classifications): the resultant confusion matrix (size: $l \times m$).

-specificity_sensitivity (only available for binary classifications): A matrix (size: $m \times 2$) stores the final specificity (the 1st column) and sensitivity (the 2nd column) after fusion. Each row

corresponds to a given Feature Selection.

-confusion_matrix (only available for multi-class classifications): The confusion matrix (size: $l \times m$) after fusion.

-accuracy_each_class (only available for multi-class classifications): A matrix with the size of $m \times l$ stores the mean accuracy of each class in cross validation.

-pred_label_vote: A matrix with the size of $n \times m$ (n is the number of samples and m is the number of Feature Selections) stores the final decisions of all samples; each column contains the final decisions of all samples after fusion for a given Feature Selection.

-pred_label: A matrix stores the decision value of all separate models. ① For classification, the size of the matrix is $n \times m \times b \times f$, where n is the number of samples, m is the number of Feature Selections, $b = C_l^2$ for SVM or $b = l$ for other machine learning algorithms (l is the number of class), and f is the number of separate models. ② For regression, size is $n \times m \times f$.

-true_label: A column vector (size: $1 \times n$) stores the true labels of all samples.

-Accuracy_vote (only available for classifications): A row vector (size: $1 \times m$) stores the final classification accuracy after decision fusion for a given Feature Selection.

-r_vote (only available for regression): A row vector (size: $1 \times m$) stores the Pearson's correlation coefficient after decision fusion for a given Feature Selection.

-p_vote (only available for regression): A row vector (size: $1 \times m$) stores the P-value (i.e., statistical significance) of the obtained Pearson's correlation coefficients for a given Feature Selection after decision fusion.

Note: If “Feature Selections” is not specified, the above dimensions related to m will be removed or set to 1.

(2) ConfusionMatrix_ m .bmp/.fig is the confusion matrix of fusion results (see Figure 3.6.2.3-1), only available for multi-class classifications.

3.7 Exit MVPANI

To exit MVPANI, simply close the main interface window by clicking the "X" button at the upper right corner.

4 Examples

In this chapter, we illustrate step-by-step how to apply MVPA using MVPANI through four examples. The data used in these four examples are MRI data from 20 patients and 20 controls.

More details can be found in Peng et al. (Frontiers in Neuroscience, 2020, 14:545). Briefly, three functional imaging metrics, including regional homogeneity (ReHo), fractional amplitude of low-frequency fluctuations (fALFF), and whole-brain functional connectivity (FC) were derived from the preprocessed fMRI data of each subject.

4.1 Example 1: Classification of Patients and Healthy Controls

In this example, we will use SVC to classify patients from controls based on whole-brain ReHo maps.

4.1.1 Launch MVPANI and Select Classification

1. Type MVPANI in the MATLAB Command Window
2. Select "Classification" mode (Figure 4.1.1-1)

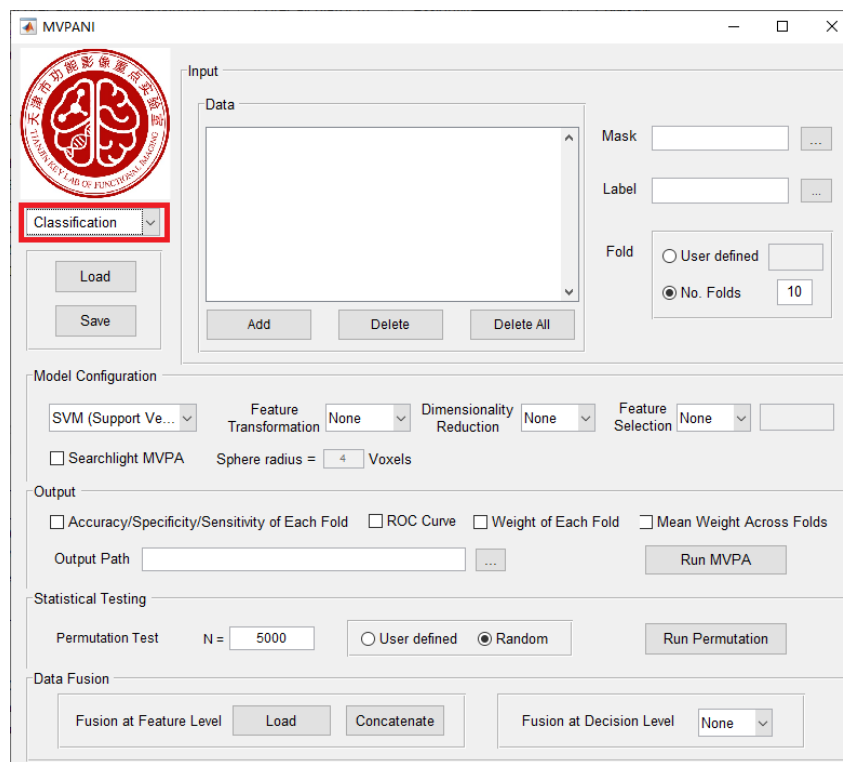


Figure 4.1.1-1 Select "Classification" mode.

4.1.2 Data Preparation and Input

4.1.2.1 ReHo Maps

1. Create a folder on your disk, assuming the path is 'G:\MVPANI_Examples\ReHo\Data'.

- Copy the ReHo maps of the 20 patients (zReHoMap_P*.nii) and 20 normal controls (zReHoMap_N*.nii) to this folder (see Figure 4.1.2.1-1).

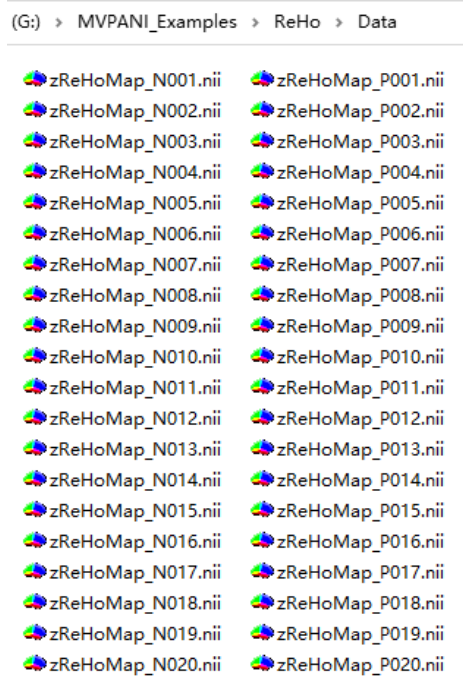


Figure 4.1.2.1-1 The folder containing the ReHo maps of 20 patients (zReHoMap_P*.nii) and 20 normal controls (zReHoMap_N*.nii).

- Add these ReHo maps in MVPANI using the "Add" button of the Input Module. Make sure the file filter is in "*.nii" format, select all files and click "Open" (Figure 4.1.2.1-2).

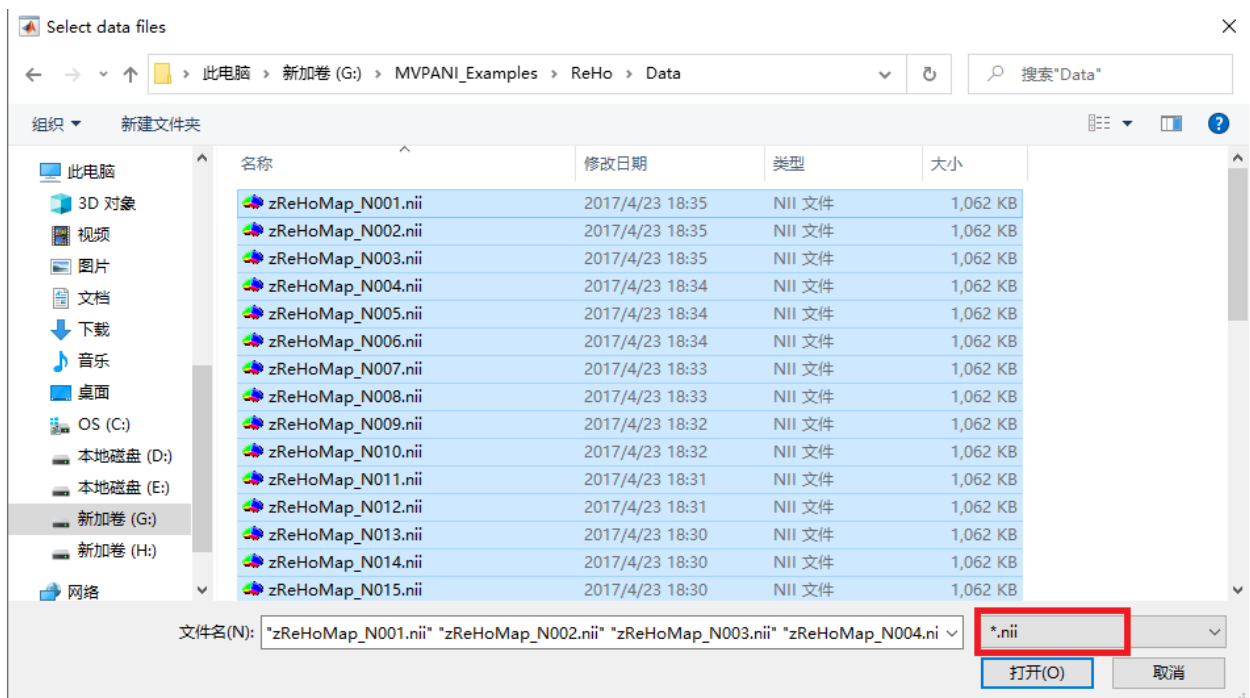


Figure 4.1.2.1-2. The popup window for adding all ReHo maps.

4. Check the order of the files loaded in MVPANI and you can see that the ReHo maps of the 20 normal controls are loaded first and followed by the ReHo maps of the 20 patients (Figure 4.1.2.1-3). The first line "(zReHoMap_N001.nii) ID:001\Tol:40" means that 40 data files in total are loaded and the file zReHoMap_N001.nii is the first.

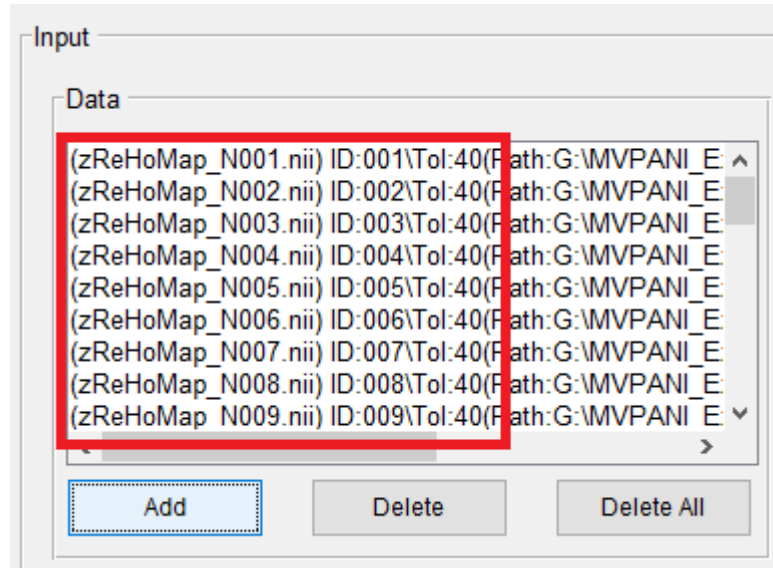


Figure 4.1.2.1-3. Check the order of the files based on the information shown in the Data box of the Input Module.

4.1.2.2 Mask File

In this example, we will use only the grey matter voxels. Therefore, we need a binary mask to include only the grey matters in the brain. In this mask, 1 indicates grey matter voxels and 0 indicates non-grey matter voxels (Figure 4.1.2.2-1). Assuming that such a grey matter mask file is in 'F:\MVPA\data\Mask'. Load this mask file to MVPANI by clicking the button beside "Mask" box in the Input Module (Figure 4.1.2.2-2).

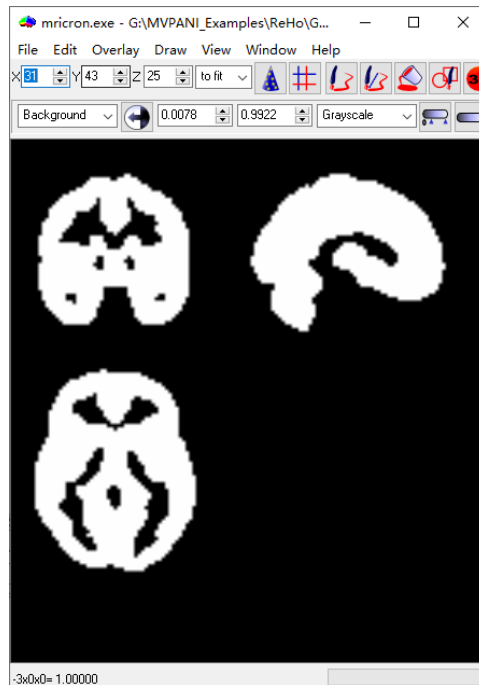


Figure 4.1.2.2-1. The grey matter mask include only the grey matter voxels.

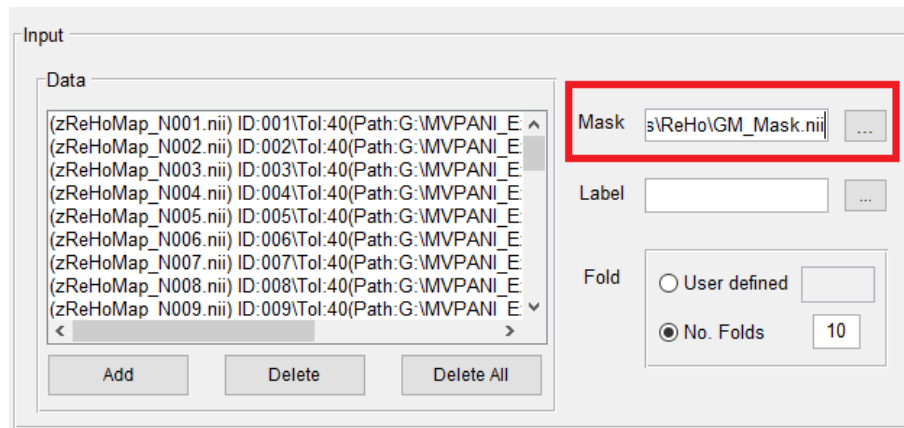


Figure 4.1.2.1-2. Load the grey matter mask file by clicking the button beside "Mask" box in the Input Module.

4.1.2.3 Label File

1. Create a Label file (Label.xlsx) in the path G:\MVPANI_Examples\ReHo using Excel. This file contains 40 numbers as a column: the first 20 numbers are -1 (correspond to normal controls) and the second 20 numbers are 1 (correspond to patients), as shown in Figure 4.1.1-2.

	A	B	C
1	1		
2	1		
3	1		
4	1		
5	1		
6	1		
7	1		
8	1		
9	1		
10	1		
11	1		
12	1		
13	1		
14	1		
15	1		
16	1		
17	1		
18	1		
19	1		
20	1		
21	-1		
22	-1		
23	-1		
24	-1		
25	-1		
26	-1		
27	-1		
28	-1		
29	-1		
30	-1		
31	-1		
32	-1		
33	-1		
34	-1		
35	-1		
36	-1		
37	-1		
38	-1		
39	-1		
40	-1		

Figure 4.1.2.3-1 The Excel file of the class labels of all samples.

2. Load this Label file to MVPANI by clicking the button beside "Label" box in the Input Module (Figure 4.1.2.3-2).

The screenshot shows the 'Input' module window. On the left, a list of data files is shown, each with a path to the MVPANI_E directory. On the right, there are fields for 'Mask' and 'Label', both with browse buttons. The 'Label' field is highlighted with a red rectangle. Below these fields, there is a 'Fold' section with two radio buttons: 'User defined' and 'No. Folds' (which is selected). A text box next to 'No. Folds' contains the number '10'.

Figure 4.1.2.3-2. Load the label file by clicking the button beside "Label" box in the Input Module.

4.1.2.4 Fold File

In this example, we will divide all subjects into 20 folds, and each fold contains a patient and a normal control.

1 Create a Fold file (Fold.xlsx) in the path G:\MVPANI_Examples\ReHo using Excel. This file contains 40 numbers as a column: the first 20 numbers are 1, 2, 3, ..., 20 (correspond to normal controls) and the second 20 numbers are also 1, 2, 3, ..., 20 (correspond to patients), as shown in Figure 4.1.2.4-1. In this way, the first control (zReHoMap_N001.nii) and the first patient (zReHoMap_P001.nii) belongs to Fold 1, the second control (zReHoMap_N002.nii) and the second patient (zReHoMap_P002.nii) belongs to Fold 2, and so on.

	A	B	C
1	1		
2	2		
3	3		
4	4		
5	5		
6	6		
7	7		
8	8		
9	9		
10	10		
11	11		
12	12		
13	13		
14	14		
15	15		
16	16		
17	17		
18	18		
19	19		
20	20		
21	1		
22	2		
23	3		
24	4		
25	5		
26	6		
27	7		
28	8		
29	9		
30	10		
31	11		
32	12		
33	13		
34	14		
35	15		
36	16		
37	17		
38	18		
39	19		
40	20		

Figure 4.1.2.4-1 The Excel file of the fold information of all samples (i.e., which fold each sample belongs to).

2 Load this Fold file to MVPANI by clicking the option "User defined" "Label" box in the Input Module (Figure 4.1.2.4-2).

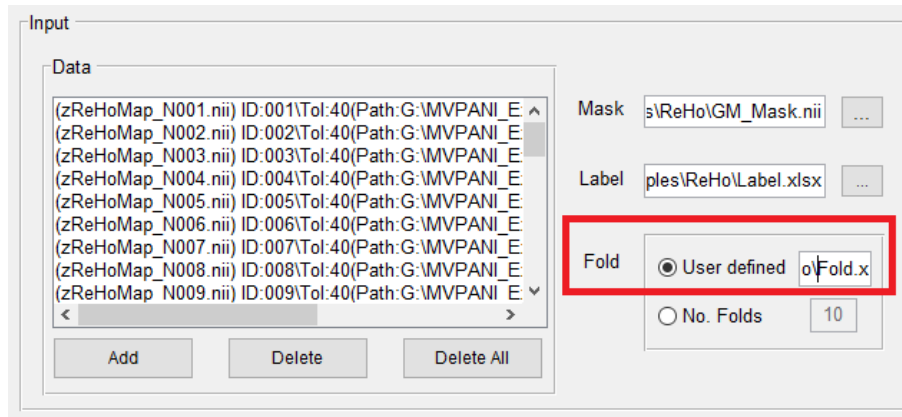


Figure 4.1.2.4-2. Load the Fold file by clicking the option "User defined" in the Input Module.

4.1.3 Model Configuration

In this example, we will use all default configurations in the Model Configuration Module, that is, use SVM, and leave Feature Transformation, Dimensionality Reduction, and Feature Selection as "None". Therefore, you can simply skip this module.

4.1.4 Output Specification and Run MVPA

We would like to obtain the classification accuracy, specificity and sensitivity of each fold, the AUC and the ROC curve, the feature weights of each fold, and the mean weights across folds.

1. Check all four options: "Accuracy, Specificity and Sensitivity of Each Fold", "ROC Curve", "Weight of Each Fold", and "Mean Weights Across Folds".
2. Create a folder on your disk to store all output files, e.g., G:\MVPANI_Examples\ReHo\Results.
3. Select this folder as the Output Path by clicking the button beside the Output Path box. (Figure 4.1.4-1)

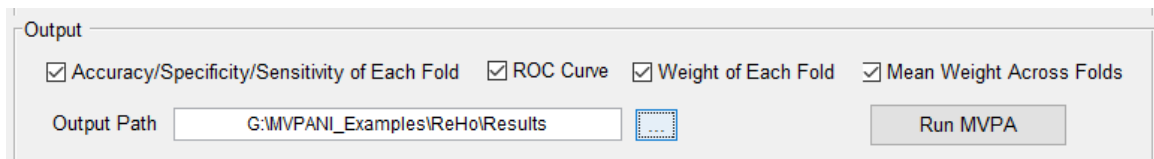


Figure 4.1.4-1 Output settings

4. Click "Run MVPA" button to start. You will see a progress bar indicating the current analysis progress and some real-time information in the Command Window. (Figure

4.1.4-2)

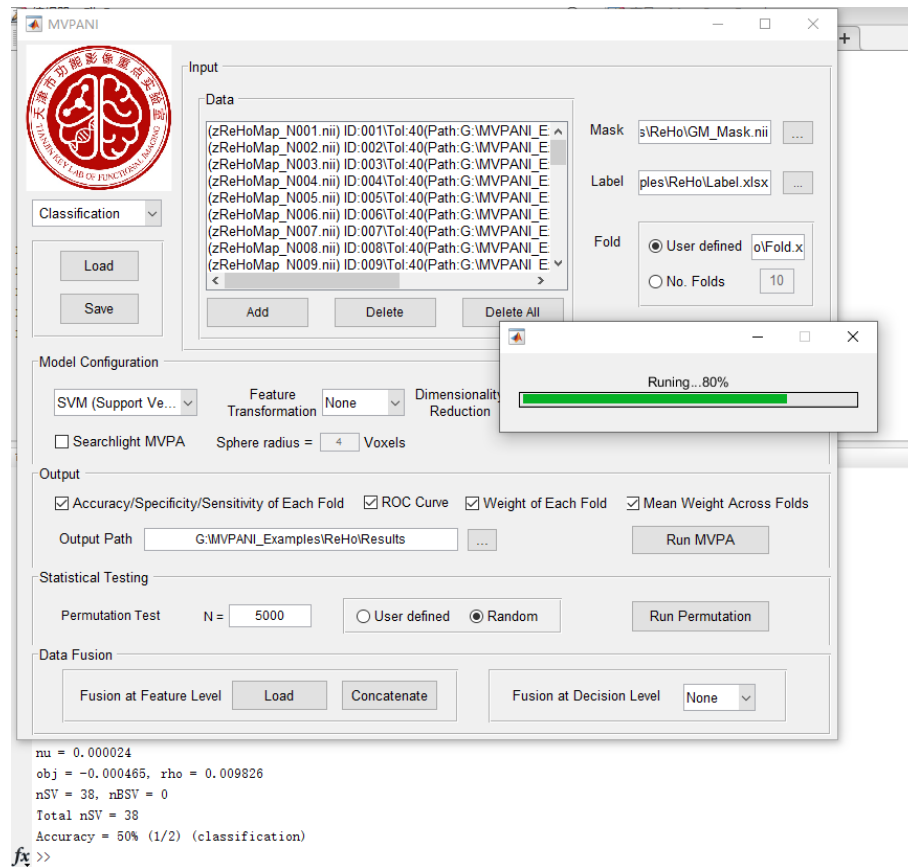


Figure 4.1.4-2 Progress bar

When the analysis is finished, a figure showing the ROC curve with AUC value (0.9125) will pop up, and the mean classification accuracy across all folds (87.5%) will be shown in the Command Window (Figure 4.1.4-3).

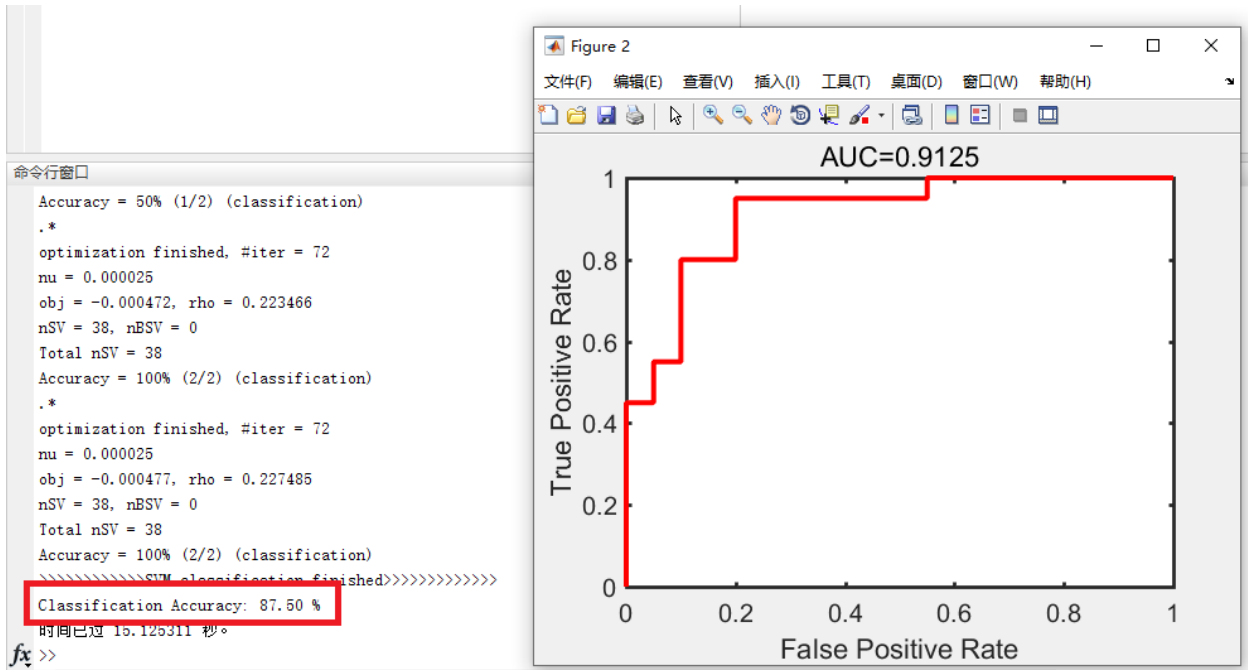


Figure 4.1.4-3 Classification accuracy (left) and the ROC curve and AUC (right).

You will also find all the output files in your Output Path "G:\MVPANI_Examples\ReHo\Results" (Figure 4.1.4-4), including the major results saved in MvpaResults.mat (Figure 4.1.4-5), the raw data of all samples after Mask filtering saved in RawData.mat, the ROC curve saved in both bmp and fig formats, the mean weights saved in both WeightImgFeature.mat and in WeightImgFeature.nii, and the feature weights obtained for each fold saved in WeightImgFeatureFold_1.mat, WeightImgFeatureFold_2.mat, etc.

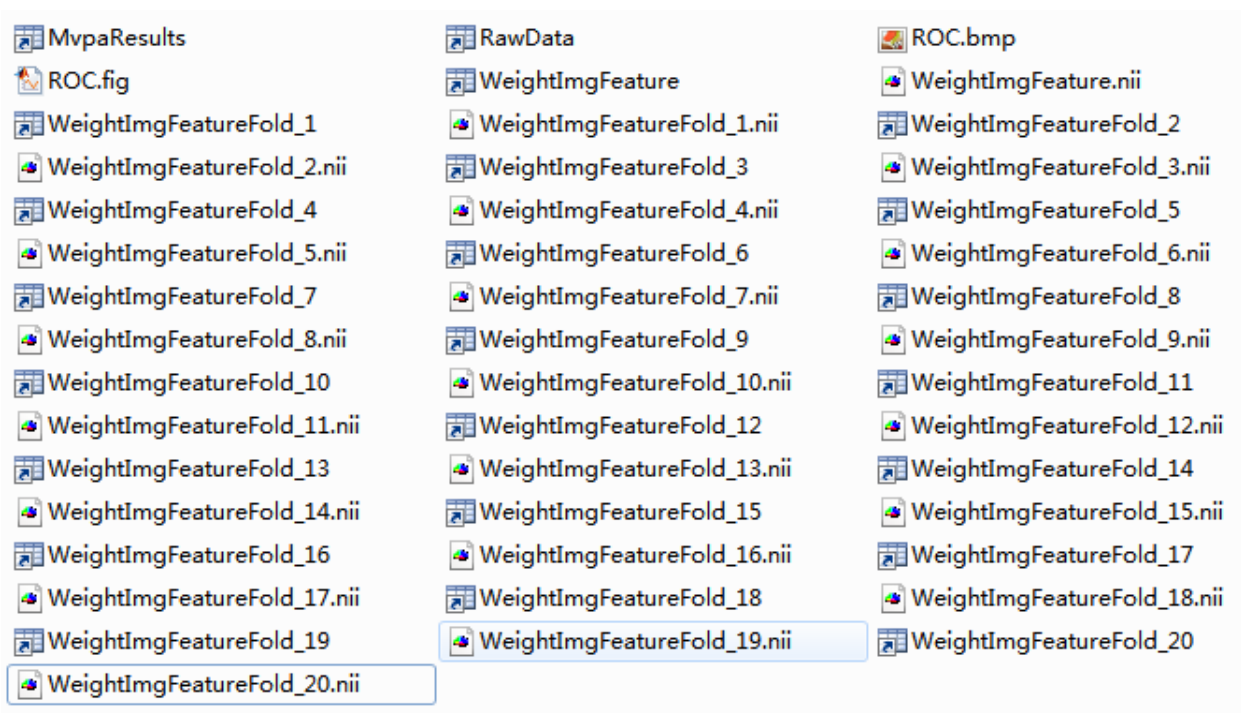


Figure 4.1.4-4 Output files

MvpaResults	
1x1 struct 包含 17 个字段	
字段	值
FeatureNum	54588
IndexColumn	1x54588 double
Fold	1x1 struct
MaskPath	'G:\MVPANI_Examples\ReHo\GM_Mas...
Label	1x1 struct
Method	'Classification'
SelectFeature	'None'
FeatureTransformation	'None'
ModelParameter	'-s 0 -t 0 -c 1 -g 0.1 -n 0.5 -r 0 -p 0.1'
AUC	0.9125
MeanAccuracy	87.5000
SpecificitySensitivity	[80,95]
Predict_test_Label	40x3 double
ClassAlgorithm	'SVM (Support Vector Machine)'
DimensionalityReduction	'None'
Specif_Sensit_acc_fold	20x3 double
GroupFile	1x40 cell

Figure 4.1.4-5 Example of MvpaResults.mat

4.1.5 Statistical Test to determine the significance of the classification accuracy

1. Set the number of permutations to 100 (here we use only 100 permutations to save the computation time) by enter 100 in the "N=" box.
2. Leave the default option "Random" as it is to let MVPANI automatically permute the training samples in a random manner.
3. Run permutation by clicking the "Run permutation" button.
4. Enter the number of CPU cores you want to use for running this permutation test (here, we enter 4). See Figure 4.1.5-1.

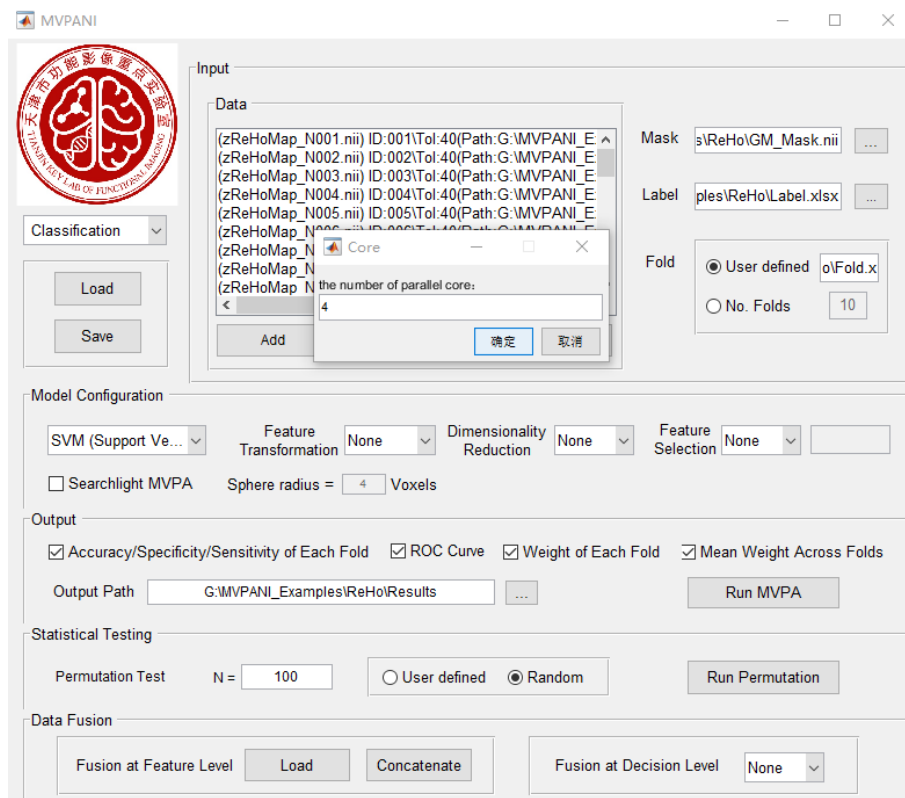


Figure 4.1.5-1 Example of permutation test configuration.

When the permutation test is finished, you will see the following histogram showing the null distribution and the actual accuracy (the red vertical line), and the P value at the top "P<0 (1/100)":

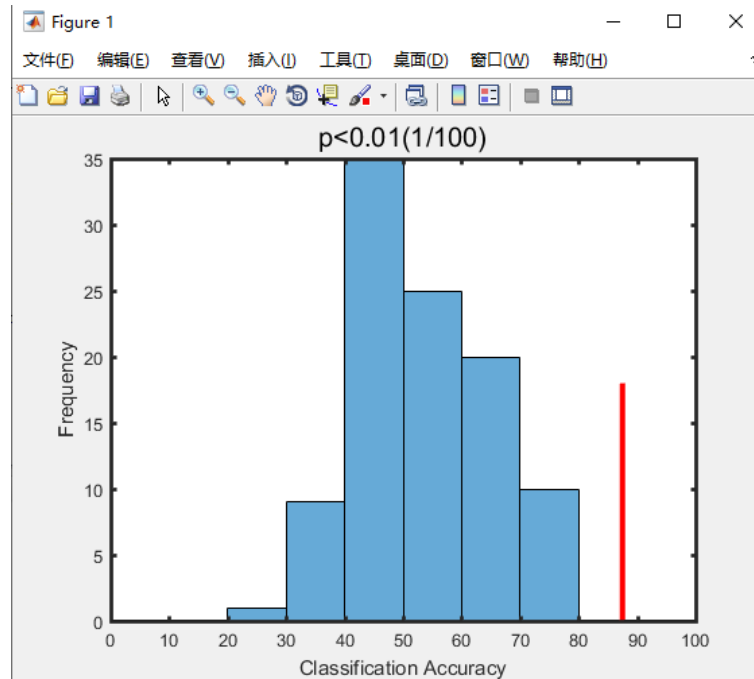


Figure 4.1.5-2 Null distribution of the permutation test (indicated by the histogram) and the actual classification accuracy (indicated by the red vertical line). The P value is shown at the top of the figure.

You will also see four new output files (Figure 4.1.5-3) are saved in the Output Path.

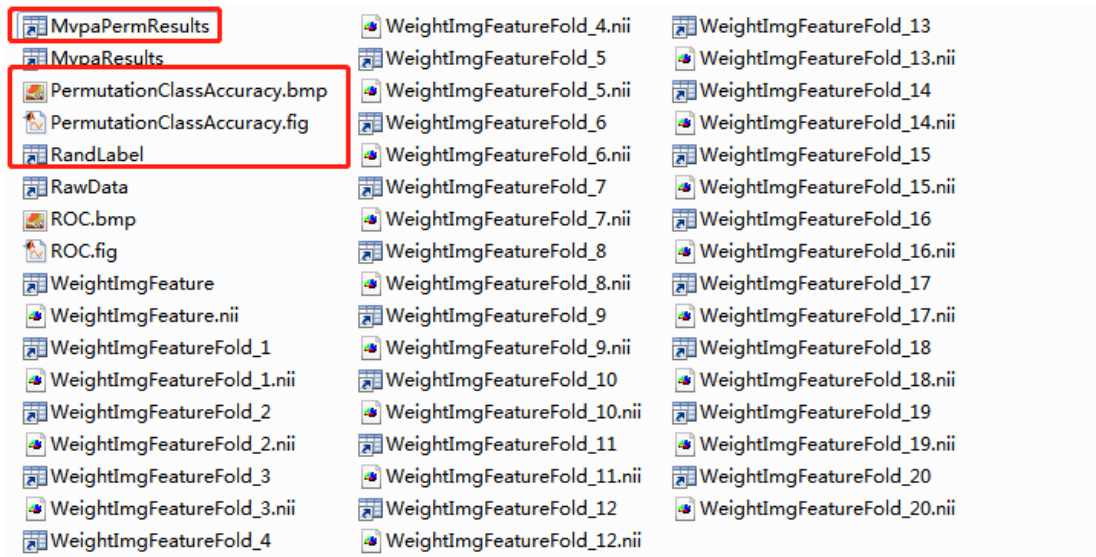


Figure 4.1.5-3 Output files of permutation test

4.2 Example 2: Classification of Patients and Healthy Controls Using Feature Selection Function

In this example, we will still use SVC to classify patients from controls based on whole-brain ReHo maps but this time we will use the Feature Selection function by setting a series of percentages of top features.

The overall processing steps of Example 2 (without statistical test) are shown in Figure 4.2-1.

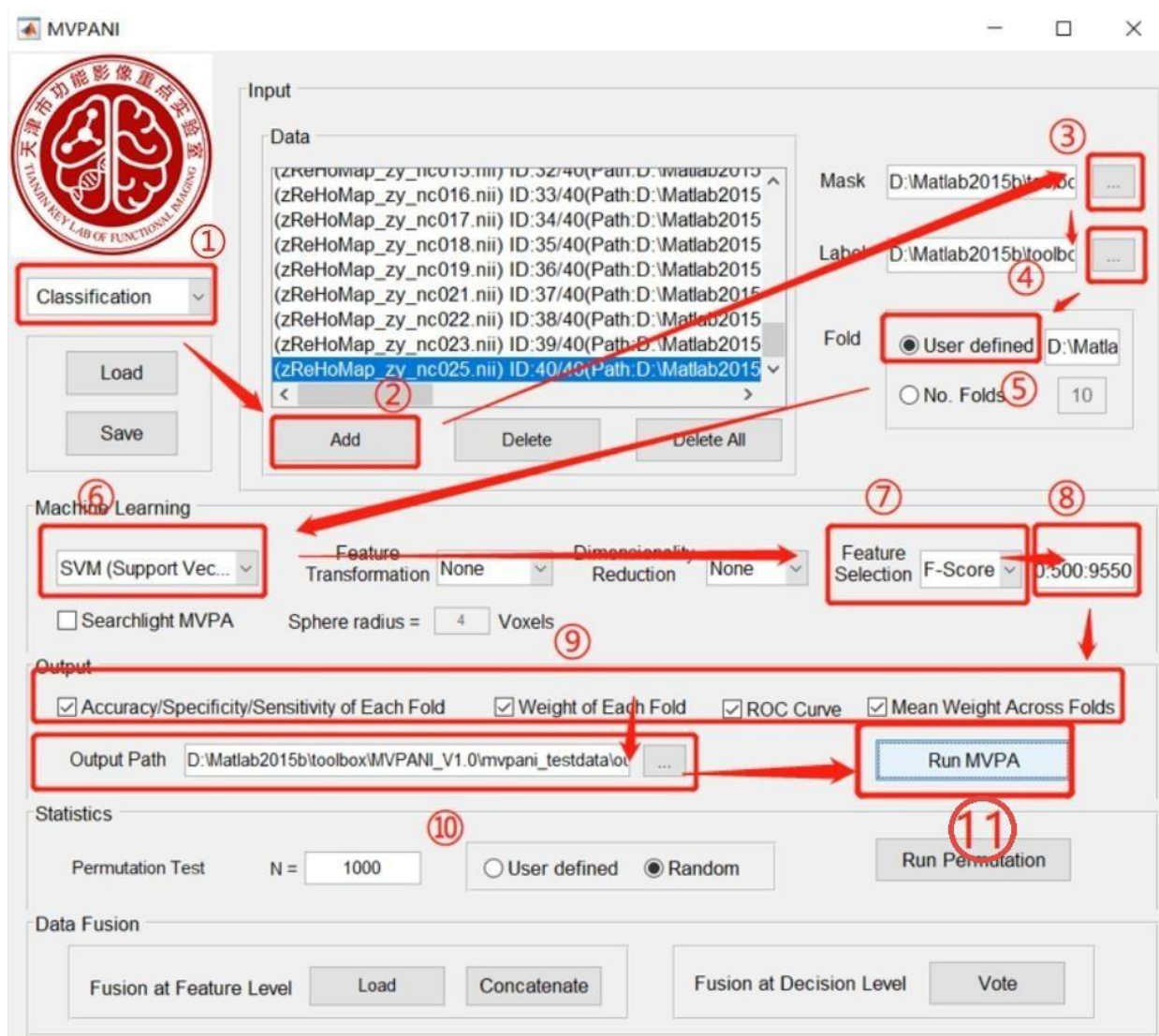


Figure 4.2-1 The processing steps in Example 2.

4.2.1 Launch MVPANI and Select Classification

Same as in Example 1. See Step ① in Figure 4.2-1.

4.2.2 Data Preparation and Input

Same as in Example 1. See Steps ② to ⑤ in Figure 4.2-1.

4.2.3 Model Configuration

See Steps ⑥ to ⑧ in Figure 4.2-1.

Use Feature Selection function to select a subset of features based on F scores for building machine learning models. First, click button ⑦ in Figure 4.2-1 and select “F-score”, and enter "50:500:9550" in the text box in Figure 4.2-1 (i.e., a series of feature selection will be performed: the first one select only 50 voxels with the highest F scores, the second one select 550 voxels with the highest F scores, and so on, until the last one select 9550 voxels with the highest F scores). The complete process is shown in Figure 4.2-1.

We chose C-SVC with default parameter settings (linear kernel; penalty coefficient $c=1$) as the classification algorithm.

4.2.4 Output Specification and Run MVPA

The output configurations are the same as in Example 1. See Steps ⑨ to ⑪ in Figure 4.2-1.

You will find all the output files in your Output Path (Figure 4.1.4-1), including the major results saved in `MvpaResults.mat`, the raw data of all samples after Mask filtering saved in `RawData.mat`, the ROC curve saved in both `bmp` and `fig` formats, the mean result of feature selection in each fold saved in `SelectFeaturePercentageImg_16.mat/nii`, etc., and the feature selection result obtained for each fold in each feature group saved in `SelectFeatureImg_1_Fold_8.mat/nii`, the mean weights of each feature group saved in `WeightImgSelectFeature_1.mat/nii`, and the feature weights obtained for each fold and each feature group saved in `WeightImgSelectFeature_1_Fold_1.mat`, `WeightImgFeature_1_Fold_2.mat`, etc.

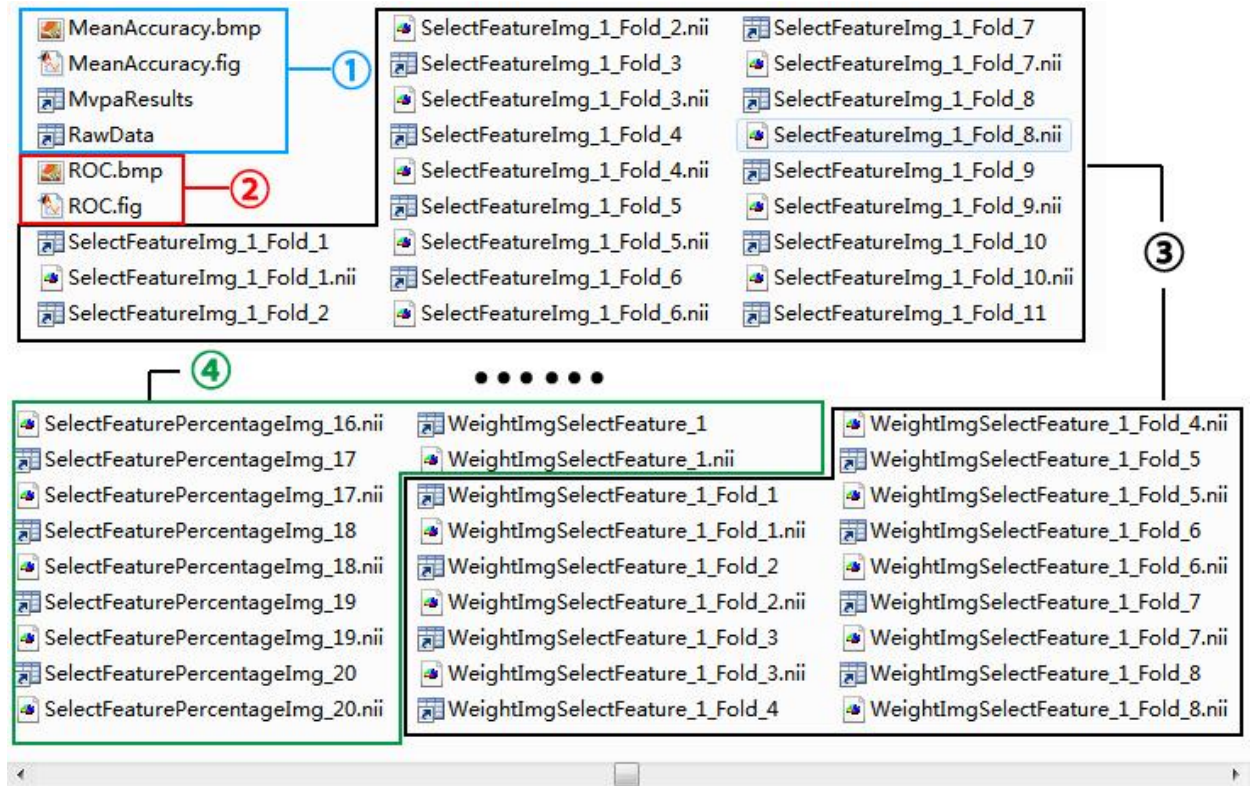


Figure 4.2.4-1 The content of the output folder in Example 2. ① main classification results. ② ROC curve. ③ Outputs for the option “Feature Weight of Each Fold”. ④ Outputs for the option “Mean Weight Across Folds”.

4.2.5 Statistical Test to Determine the Significance of the Classification Accuracy

The operation steps are the same as in Example 1.

Figure 4.2.5-1 shows the results of the permutation test (a), together with the obtained ROC curve (b), classification accuracy of each feature selection group (c), and feature weight map (d). Except for the classification accuracies, other results are shown only for the feature selection group with the highest accuracy.

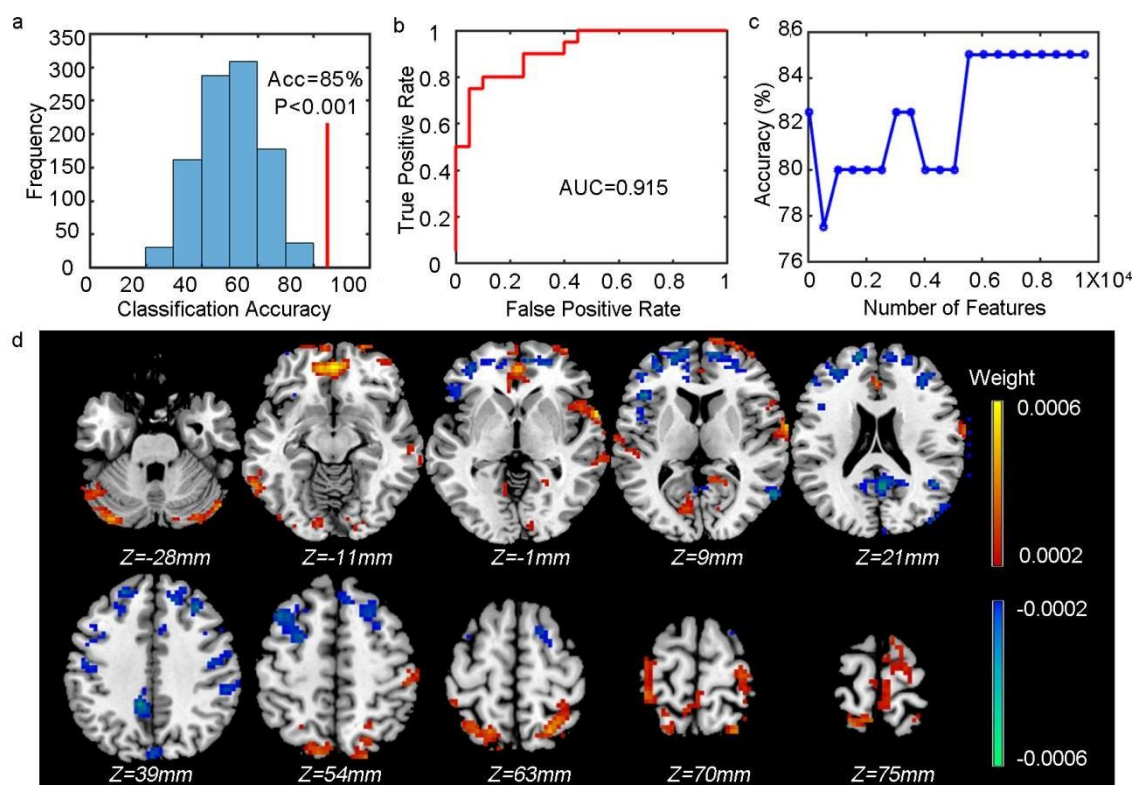


Figure 4.2.5-1 Classification results of Example 2. Panel a, the P-value obtained from the null distribution generated by 1,000 permutations (indicated by the blue histogram) and the actual classification accuracy (Acc, indicated by the red vertical line) obtained when 60% features were selected which resulted in the highest AUC 0.915. Panel b, the ROC curve and its AUC obtained for the feature selection with the highest AUC. Panel c, the classification accuracy as a function of the number of selected features. Panel d, the feature weight map (only the voxels with absolute weights >0.0002 are shown). The color of each voxel indicates the sign of its weight (blue-green indicates negative weights and red-yellow indicates positive weights).

4.3 Example 3: Information Mapping with Searchlight MVPA

In order to locate brain regions that contain sufficient information (e.g., different spatial patterns of brain activity which can distinguish patients from healthy controls) based on ReHo maps, we performed searchlight MVPA using SVC. The data used in this example are the same as in Example 1 (Section 4.1).

The overall processing steps (without statistical test) of Example 3 are shown in Figure 4.3-1.

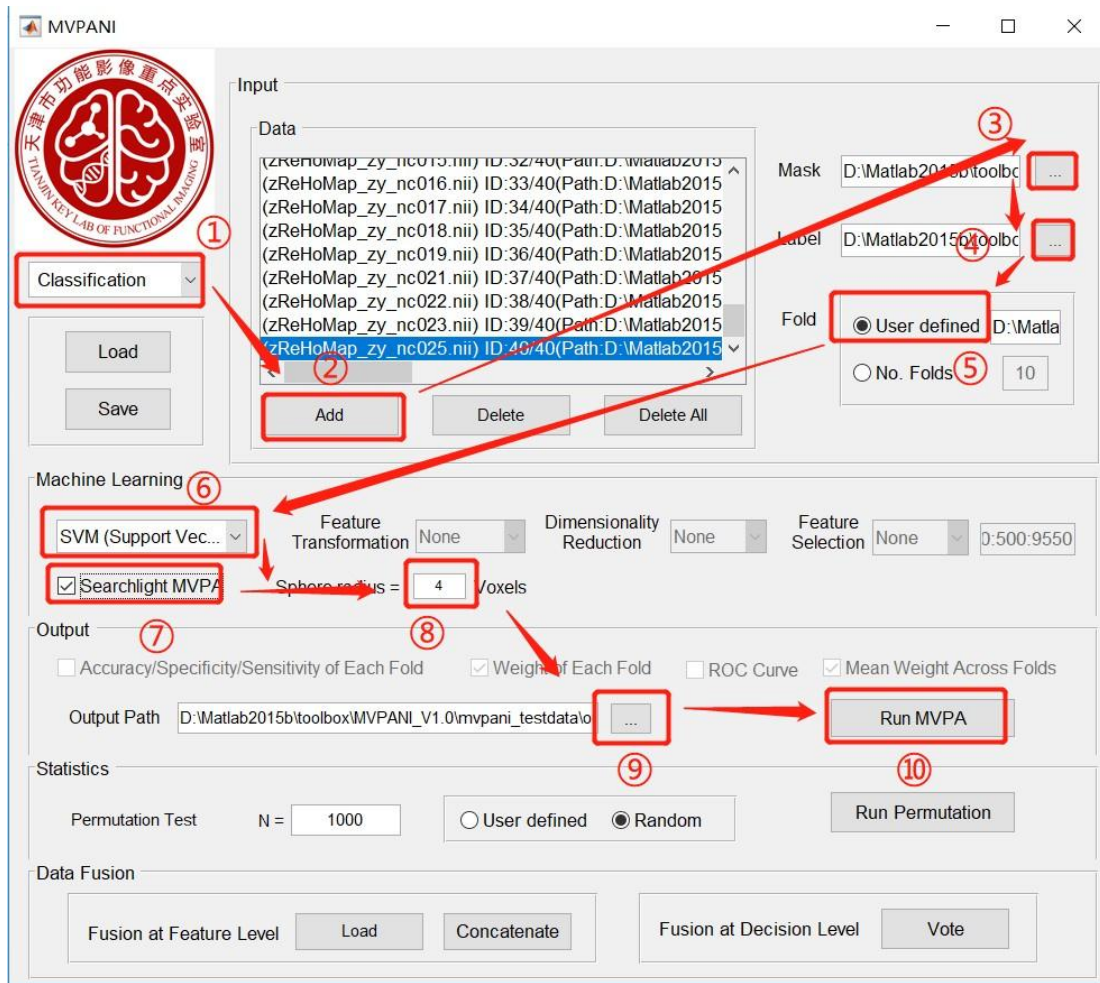


Figure 4.3-1 Processing steps in Example 3.

4.3.1 Data Preparation and Input

Same as in Example 1 (Sections 4.1.1 and 4.1.2).

4.3.2 Model Configuration

Select the "Searchlight MVPA" option to perform Searchlight MVPA, where the size of a searchlight area is defined as a sphere with a 4-voxel radius. We still use C-SVC with default parameter settings as the classification algorithm. No feature transformation or feature selection is applied. The overall operation flow is shown in Figure 4.3-1, and detailed below:

- ① Select the Classification mode.
- ①-⑥ Same as in Example 2.
- ⑦ Check the 'Searchlight MVPA' option.

- ⑧ Enter '4' to specify the sphere radius (i.e., 4 voxels).
- ⑨ Specify the output path.
- ⑩ Click the 'Run MVPA' button to start running.

4.3.3 Output

The main output is a classification accuracy map averaged over all cross-validation steps (as shown in Figure 4.3.3-1), i.e., the value for each voxel represents the average classification accuracy (after subtraction of the chance-level accuracy 50%) obtained using all voxels included in the predefined Searchlight sphere centered on that given voxel.

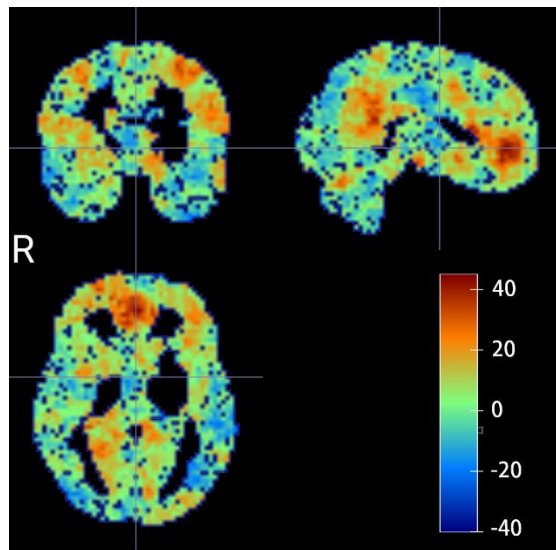


Figure 4.3.3-1 Brain map of classification accuracies obtained using Searchlight MVPA.

4.3.4 Permutation Test

To identify the voxels with classification accuracies significantly higher than the chance-level accuracy, we performed a permutation test ($N=500$). Figure 4.3.4-1 shows the output files and folders.

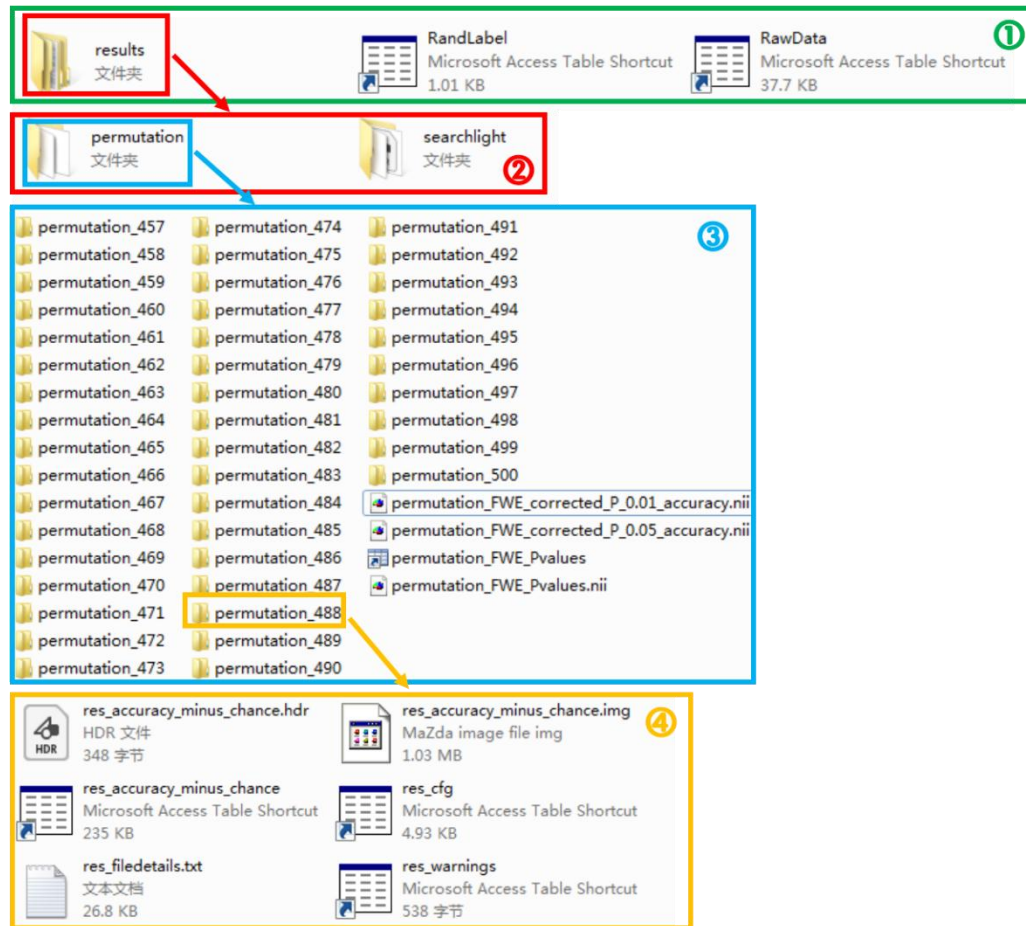


Figure 4.3.4-1 Outputs of permutation test for Searchlight MVPA. ① This folder is the output path. ② The main results are saved in the folder “permutation” which is in the “results” folder. ③ Subfolders in the “permutation” folder. Each subfolder named “permutation_ \hat{N} ” corresponds to the result of a particular permutation step. Other files are permutation P values and classification accuracies. ④ All files in a subfolder “permutation_ \hat{N} ”.

4.4 Example 4: Predicting Reaction Time Using Brain Imaging Data

In this example, we demonstrate how regression in MVPANI can be used to predict continuous values based on brain imaging data. More specifically, we use fALFF maps obtained from rs-fMRI data to predict reaction times (RT) to test whether resting-state brain activity can predict task performance.

The overall processing steps (without statistical test) of Example 4 are shown in Figure 4.4-1.

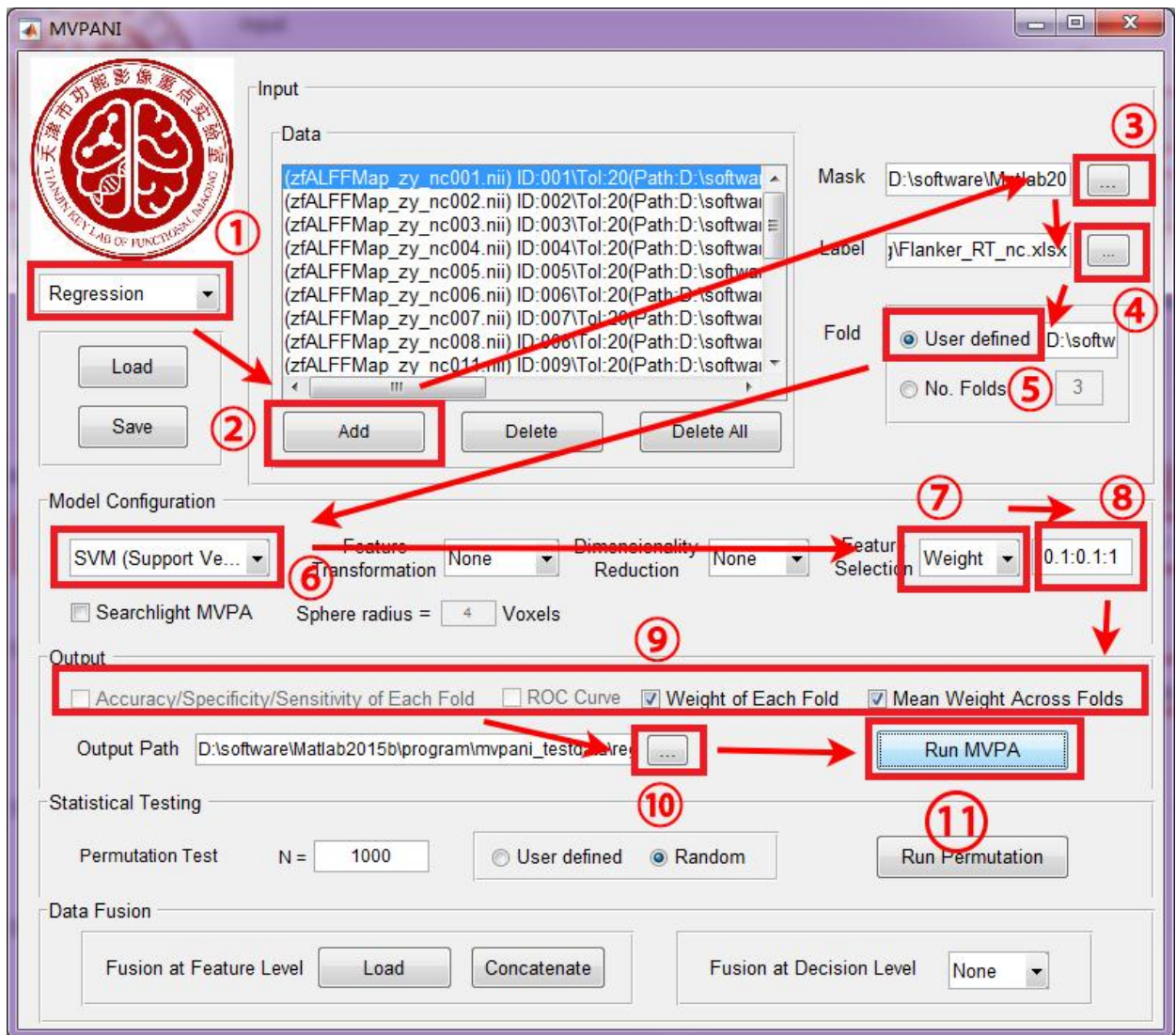


Figure 4.4.3-1 The whole operation steps in Example 4. ① Click the drop-down menu and select Regression. ② Click the 'Add' button and add all subjects' fALFF maps. ③ Click the '...' button and add the Mask file. ④ Click the '...' button to add the Label file. ⑤ Click the 'User defined' option to add the Fold file. ⑥ Click the drop-down menu, select 'SVM (Support Vector Machine)' option, and click 'Done' button in the pop-up window with the default parameters unchanged. ⑦ Click the 'Feature Selection' drop-down menu, and select the 'Weight' option. ⑧ Enter 0.1:0.1:1 (for Chinese users, note that the colon must be entered in English mode and cannot be in Chinese mode). ⑨ Check the two output options ("Weight of Each Fold" and "Mean Weight Across Folds") in the 'Output' module. ⑩ Click the '...' button to select the output path. ⑪ Click the 'Run MVPA' button to start analysis.

4.4.1 Launch MVPANI and Select Regression

1. Type MVPANI in the MATLAB Command Window
2. Select "Regression" mode (Figure 4.4.1-1)

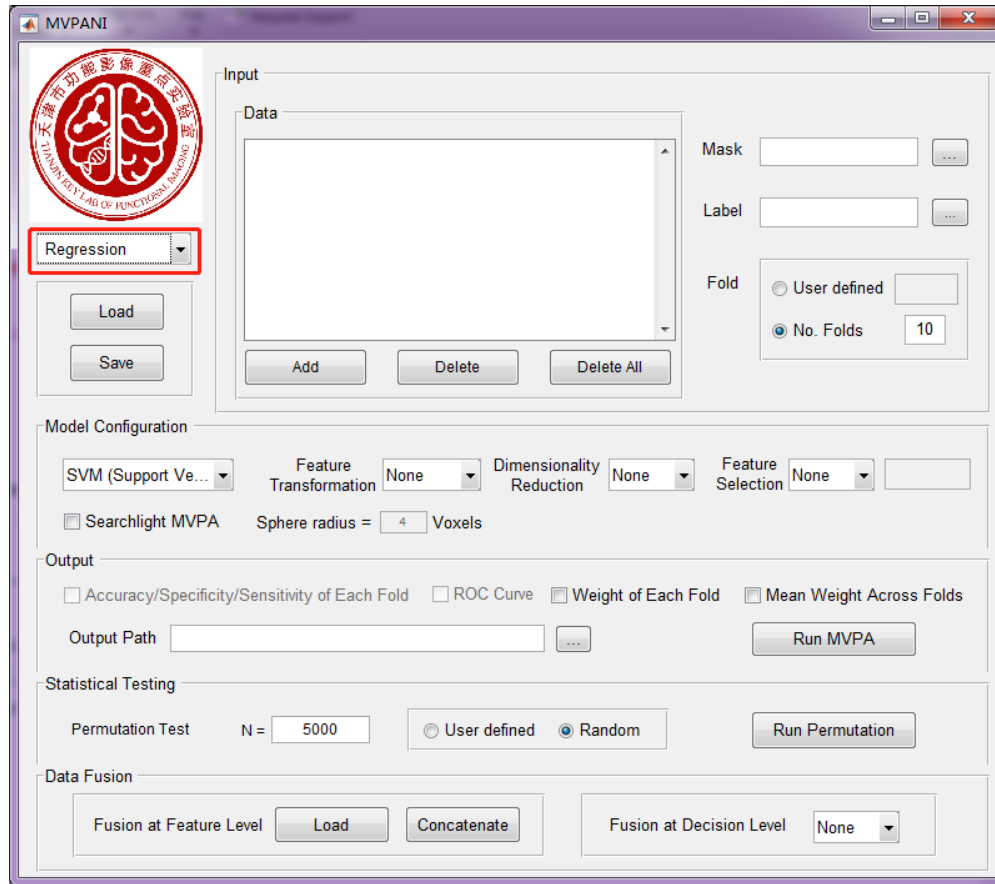


Figure 4.4.1-1 Select “Regression” mode for reaction time prediction in Example 4.

4.4.2 Data Preparation and Input

4.4.2.1 fALFF Maps and Task Reaction Times

The operation steps are the same as in Example 1, but here the input data are the fALFF maps of 20 healthy subjects (as in Figure 4.4.2.1-1). Reaction times (RTs) were averaged across all trials to represent the task performance of a given participant.

名称	修改日期	类型	大小
zfALFFMap_zy_nc001.nii	2017/4/23 18:06	Nii 文件	1,062 KB
zfALFFMap_zy_nc002.nii	2017/4/23 18:05	Nii 文件	1,062 KB
zfALFFMap_zy_nc003.nii	2017/4/23 18:05	Nii 文件	1,062 KB
zfALFFMap_zy_nc004.nii	2017/4/23 18:05	Nii 文件	1,062 KB
zfALFFMap_zy_nc005.nii	2017/4/23 18:05	Nii 文件	1,062 KB
zfALFFMap_zy_nc006.nii	2017/4/23 18:05	Nii 文件	1,062 KB
zfALFFMap_zy_nc007.nii	2017/4/23 18:04	Nii 文件	1,062 KB
zfALFFMap_zy_nc008.nii	2017/4/23 18:04	Nii 文件	1,062 KB
zfALFFMap_zy_nc011.nii	2017/4/23 18:04	Nii 文件	1,062 KB
zfALFFMap_zy_nc012.nii	2017/4/23 18:04	Nii 文件	1,062 KB
zfALFFMap_zy_nc014.nii	2017/4/23 18:03	Nii 文件	1,062 KB
zfALFFMap_zy_nc015.nii	2017/4/23 18:03	Nii 文件	1,062 KB
zfALFFMap_zy_nc016.nii	2017/4/23 18:03	Nii 文件	1,062 KB
zfALFFMap_zy_nc017.nii	2017/4/23 18:03	Nii 文件	1,062 KB
zfALFFMap_zy_nc018.nii	2017/4/23 18:02	Nii 文件	1,062 KB
zfALFFMap_zy_nc019.nii	2017/4/23 18:02	Nii 文件	1,062 KB
zfALFFMap_zy_nc021.nii	2017/4/23 18:02	Nii 文件	1,062 KB
zfALFFMap_zy_nc022.nii	2017/4/23 18:02	Nii 文件	1,062 KB
zfALFFMap_zy_nc023.nii	2017/4/23 18:01	Nii 文件	1,062 KB
zfALFFMap_zy_nc025.nii	2017/4/23 18:01	Nii 文件	1,062 KB

Figure 4.4.2.1-1 The fALFF images of 20 healthy subjects.

4.4.2.2 Mask File

Same as in Example 1.

4.4.2.3 Label File

The operation steps are the same as in Example 1. The label file contains RT values (as in Figure 4.4.2.3-1) which is a column of 20 continuous values, each corresponding to the average RT of a participant. The order of the participants in the label file corresponds to the order of the input fALFF maps.

	A	B	
1	607.43		
2	540.48		
3	421.79		
4	572.92		
5	517.47		
6	567.23		
7	422.59		
8	575.02		
9	538.17		
10	539.37		
11	569.43		
12	539.2		
13	477.49		
14	487.87		
15	463.81		
16	503.36		
17	527.75		
18	590.32		
19	495.03		
20	460.32		
21			
22			

Figure 4.4.2.3-1 Label file

4.4.2.4 Fold File

The operation steps are the same as in Example 1.

More specifically, to perform leave-one-participant-out cross-validation, the fold file (Figure 4.4.2.4-1) contains a column of "1", "2", "3" to "20", each number representing one participant.

	A	B
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
10	10	
11	11	
12	12	
13	13	
14	14	
15	15	
16	16	
17	17	
18	18	
19	19	
20	20	

Figure 4.4.2.4-1 The fold file.

4.4.3 Model Configuration

See Steps ⑥ to ⑧ in Figure 4.4-1 for model configuration. The fALFF images were used for the subsequent prediction analysis without any feature transformation. We tested how feature selection would affect prediction accuracy by selecting only the top voxels with the highest Weights. The percentage of selected voxels ranged from 10% to 100% in steps of 10% (i.e., 0.1:0.1:1). For the regression algorithm, we chose e-SVR with default parameter settings (linear kernel; penalty factor $c = 1$; ϵ ($-p$) = 0.1 in the loss function).

4.4.4 Output Specification and Run MVPA

See Steps ⑨ to ⑪ in Figure 4.4-1 for output specification.

After MVPA is finished, you will find all the output files in your Output Path, including the

major results saved in MvpaResults.mat, the raw data of all samples after Mask filtering saved in RawData.mat, the mean result of feature selection across all folds saved in SelectFeaturePercentageImg_9.mat/nii, etc., and the feature selection result obtained for each fold in each feature group saved in SelectFeatureImg_1_Fold_1.mat/nii, etc., the mean weights of each feature group saved in WeightImgSelectFeature_1.mat/nii, and the feature weights obtained for each fold and each feature group saved in WeightImgSelectFeature_1_Fold_1.mat, WeightImgFeature_1_Fold_2.mat, etc.

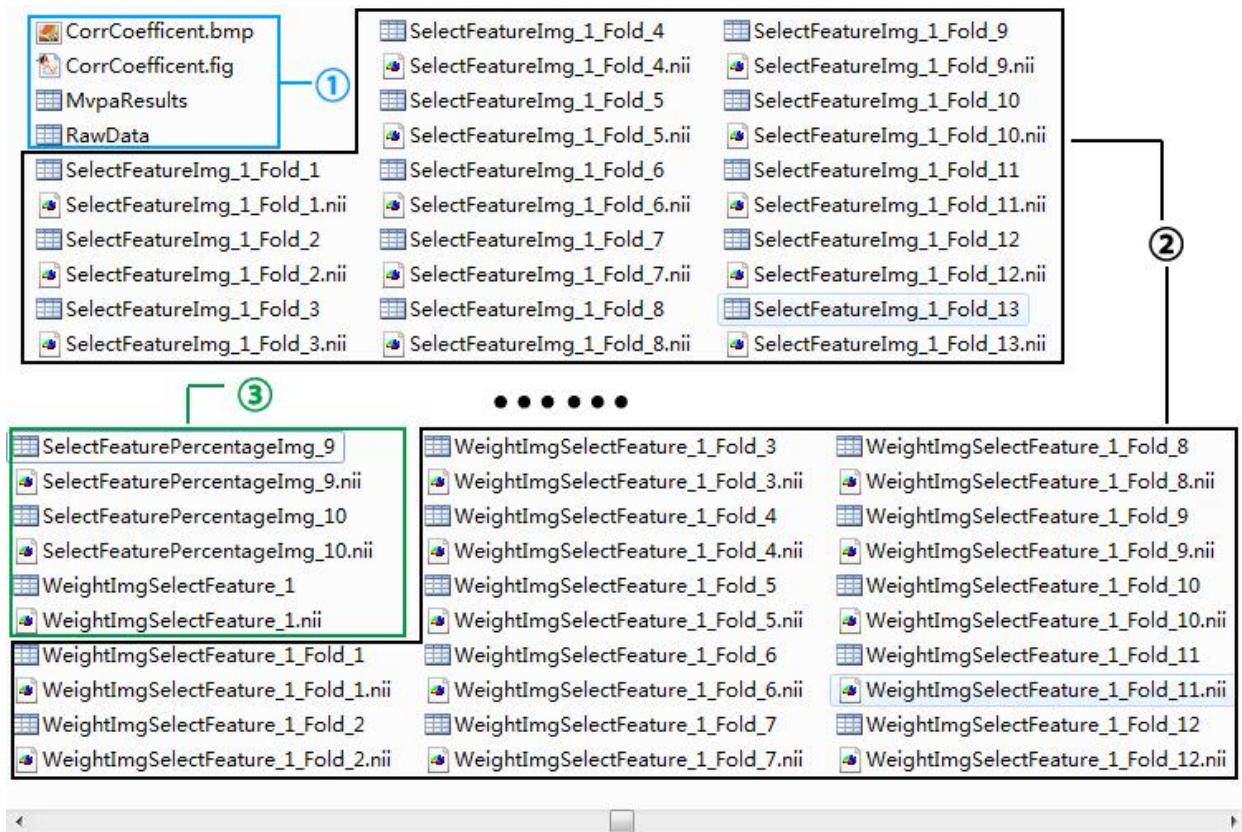


Figure 4.4.4-1 The content of the output folder in Example 4. ① Main results. ② Outputs of “Feature Weight of Each Fold”. ③ Outputs of “Mean Weight Across Folds”.

4.4.5 Statistical Test to Determine the Significance of the Correlation Coefficient Between the Predicted Values and the True Values

The operation steps are the same as in Example 1.

To test whether the highest correlation coefficient ($R=0.502$) was significantly higher than the chance level, a permutation test ($N=1000$) was performed to establish the null distribution of the chance-level correlation coefficients. A comparison between the actual correlation coefficient

obtained from the true RT and the null distribution is shown in Figure 4.4.5-1a. We found that 16 out of 1000 permutations yielded a correlation coefficient equal to or greater than the actual correlation coefficient (0.502), resulting in $P = 0.016 < 0.05$, indicating that the obtained actual correlation coefficient was significantly higher than the chance level.

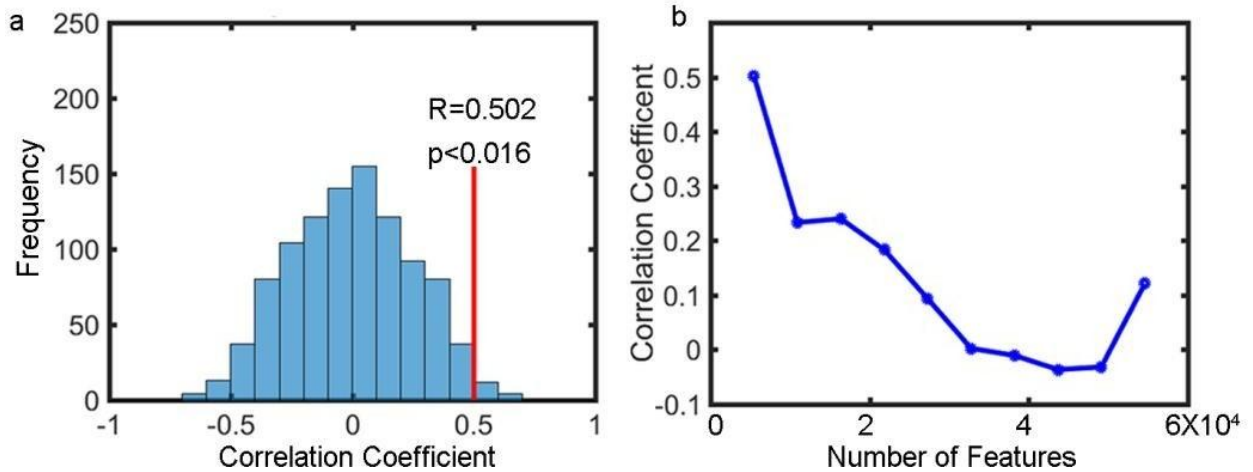


Figure 4.4.5-1 Results of reaction times (RTs) prediction. Panel a, the P-value obtained from the null distribution generated by 1,000 permutations (indicated by the blue histogram) and the actual correlation coefficient (indicated by the red vertical line) obtained when 10% features were selected which resulted in the highest correlation coefficient 0.915. Panel b, the correlation coefficient as a function of the number of selected features.

4.5 Example 5: Using Data Fusion to Improve Classification Performance

In this example, we demonstrate how to combine different measures extracted from multi-modal neuroimaging data to improve the classification performance between patients and healthy controls. We tested two "data fusion" strategies implemented in MVPANI - "Fusion at Feature Level" and "Fusion at Decision Level" - and show that both data fusion strategies improved classification performance.

4.5.1 Fusion at Feature Level

4.5.1.1 Launch MVPANI

Type MVPANI in the MATLAB Command Window.

4.5.1.2 Data Preparation

In this example, the fALFF maps in NIfTI format (as in Figure 4.5.1.2-1) and the FC matrices

in txt format (as in Figure 4.5.1.2-2) are fused for classification between patients and controls.

(1) Create a folder (e.g., named "Data_fALFF") on your disk. Copy the fALFF files of the 20 patients (zfALFFMap_zh_bg*.nii) and 20 normal controls (zfALFFMap_zy_nc*.nii) to this folder.

(2) Create a folder (e.g., named "Data_FC") on your disk. Copy the FC files of the 20 patients (zFCMap_zh_bg*.txt) and 20 normal controls (zFCMap_zh_nc*.txt) to this folder.

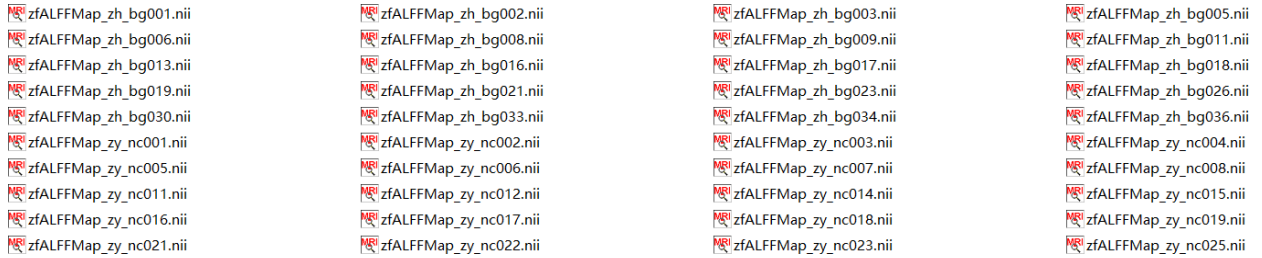


Figure 4.5.1.2-1 The fALFF data to be fused.

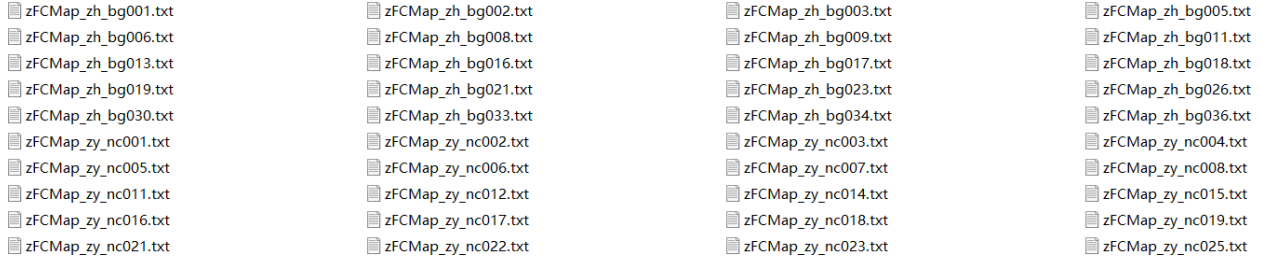


Figure 4.5.1.2-2 The FC data to be fused.

4.5.1.3 Data Fusion

Use "Load" and "Concatenate" functions in the "Data Fusion" module to select and concatenate the two types of data to form a concatenated feature vector for each subject. The operation steps are shown in Figure 4.5.1.3-1, and the generated files (containing the concatenated feature vectors) are shown in Figure 4.5.1.3-2. Specifically, click "Load" button and then add all fALFF maps and click "Done" button to load all fALFF data; then repeat "Load" - "Add" - "Done" to load all FC data; click "Concatenate" to perform the concatenation of the two types of data. Note that the order of samples (i.e., subjects) must be the same for fALFF files and the FC files when adding the data files.

In addition, a mask can be specified when selecting the files of each data type (Step ④).

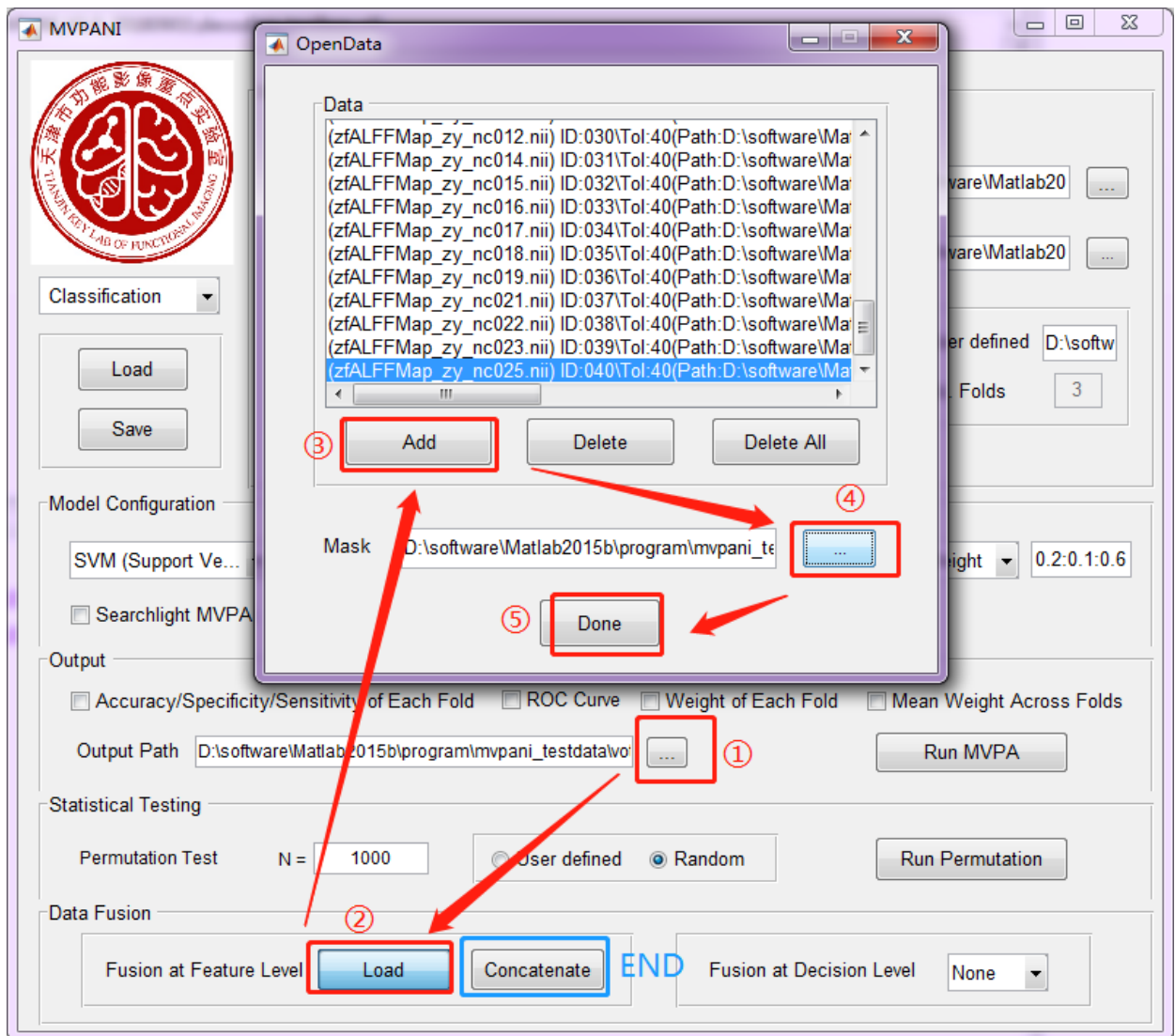


Figure 4.5.1.3-1 Schematic diagram of "Fusion at Feature Level". ① Click the '...' button to select the output path for storing the files with concatenated feature vectors that will be generated later. ② Click the 'Load' button to open the 'OpenData' window. ③ In the 'OpenData' window, click the 'Add' button to select all fALFF files. ④ Click the '...' button to select a Mask file. ⑤ Click the button 'Done' to finish loading the fALFF data. Repeat Steps ②–⑤ to load the FC data files. ⑥ Click the 'Concatenate' button to perform the concatenation of the fALFF data and the FC data and generate new files with concatenated feature vectors in the output path.

bg001_FeatureFusion.mat
bg006_FeatureFusion.mat
bg013_FeatureFusion.mat
bg019_FeatureFusion.mat
bg030_FeatureFusion.mat
nc001_FeatureFusion.mat
nc005_FeatureFusion.mat
nc011_FeatureFusion.mat
nc016_FeatureFusion.mat
nc021_FeatureFusion.mat

bg002_FeatureFusion.mat
bg008_FeatureFusion.mat
bg016_FeatureFusion.mat
bg021_FeatureFusion.mat
bg033_FeatureFusion.mat
nc002_FeatureFusion.mat
nc006_FeatureFusion.mat
nc012_FeatureFusion.mat
nc017_FeatureFusion.mat
nc022_FeatureFusion.mat

bg003_FeatureFusion.mat
bg009_FeatureFusion.mat
bg017_FeatureFusion.mat
bg023_FeatureFusion.mat
bg034_FeatureFusion.mat
nc003_FeatureFusion.mat
nc007_FeatureFusion.mat
nc014_FeatureFusion.mat
nc018_FeatureFusion.mat
nc023_FeatureFusion.mat

bg005_FeatureFusion.mat
bg011_FeatureFusion.mat
bg018_FeatureFusion.mat
bg026_FeatureFusion.mat
bg036_FeatureFusion.mat
nc004_FeatureFusion.mat
nc008_FeatureFusion.mat
nc015_FeatureFusion.mat
nc019_FeatureFusion.mat
nc025_FeatureFusion.mat

Figure 4.5.1.3-2 The new files with concatenated feature vectors in the output path.

4.5.1.4 Perform MVPA using concatenated data

The newly generated feature data are available now in the specified output path which will be used as the input data files for MVPA based on concatenated data. The operation steps for running a classification task (e.g., distinguishing patients and controls) are exactly the same as described in Example 1 (Section 4.1). The Label file and the Fold file are the same as in Section 4.1. A mask file is not needed here. For the model configuration, C-SVC with default parameter settings (as used in Section 4.1) is selected as the classification algorithm, and no Feature Transformation or Feature Selection is performed. The obtained classification accuracy is 82.25%, which is higher than the accuracy obtained from each type of features alone (80% for using fALFF maps alone and 70% for using FC matrices alone).

4.5.2 Fusion at Decision Level

In this example, we will perform classification between patients and controls by fusing the classification results obtained separately from the ReHo maps (obtained from the rs-fMRI data), the GMV maps (grey matter volumes, obtained from the structural images) and the WMV maps (white matter volumes, obtained from the structural images). To this end, we need to first obtain the classification results obtained from each type of data (Section 4.5.2.1), and then fusing the results obtained from all types of data by voting strategy to obtain the final classification result (Section 4.5.2.2). The model configuration is the same as the one used in the Section 4.5.1. We obtained 95% classification accuracy using the "voting" strategy, which is higher than the accuracy obtained from each type of features alone (85% accuracy with ReHo maps, 87.5% accuracy with GMV maps, and 92.5% accuracy with WMV maps).

4.5.2.1 Run MVPA separately using each type of data

To run MVPA separately for each type of data, the operation steps are the same as in Example 1 (Section 4.1):

First, run MVPA using the ReHo maps, and rename the result file as "ReHo_MvpaResults.mat".

Second, run MVPA using the GMV maps, and rename the result file as "GMV_MvpaResults.mat".

Third, run MVPA using the WMV maps, and rename the result file as

"WMV_MvpaResults.mat".

Fourth, create a new folder on disk (e.g., 'D:\MVPANI_V1.0\mvpani_testdata\outputdata2'), and copy the three result files (ReHo_MvpaResults.mat, GMV_MvpaResults.mat, WMV_MvpaResults.mat) to this folder.



Figure 4.5.2.1-1 The MVPA result files needed for Fusion at Decision Level in Example 5.

4.5.2.2 Fusing the three classification results

(1) Create a folder on your disk to save the final result of decision fusion, assuming the path is 'D:\MVPANI_V1.0\mvpani_testdata\outputdata2\UnweightVote'.

(2) Just follow the steps shown in Figure 4.5.2.2-1 to fuse the three classification results by voting strategy (unweighted).

The output file "UnWeightedVoteResults.mat" will be generated in the specified output path (i.e., 'D:\MVPANI_V1.0\mvpani_testdata\outputdata2\UnweightVote'). Load this file in MATLAB, you can find the final specificity and sensitivity (specificity_sensitivity), the final decisions of all samples (pred_label_vote), the decisions of all separate models (pred_label), the true labels of all samples (true_label), and the final classification accuracy (Accuracy_vote). (Figure 4.5.2.2-2).

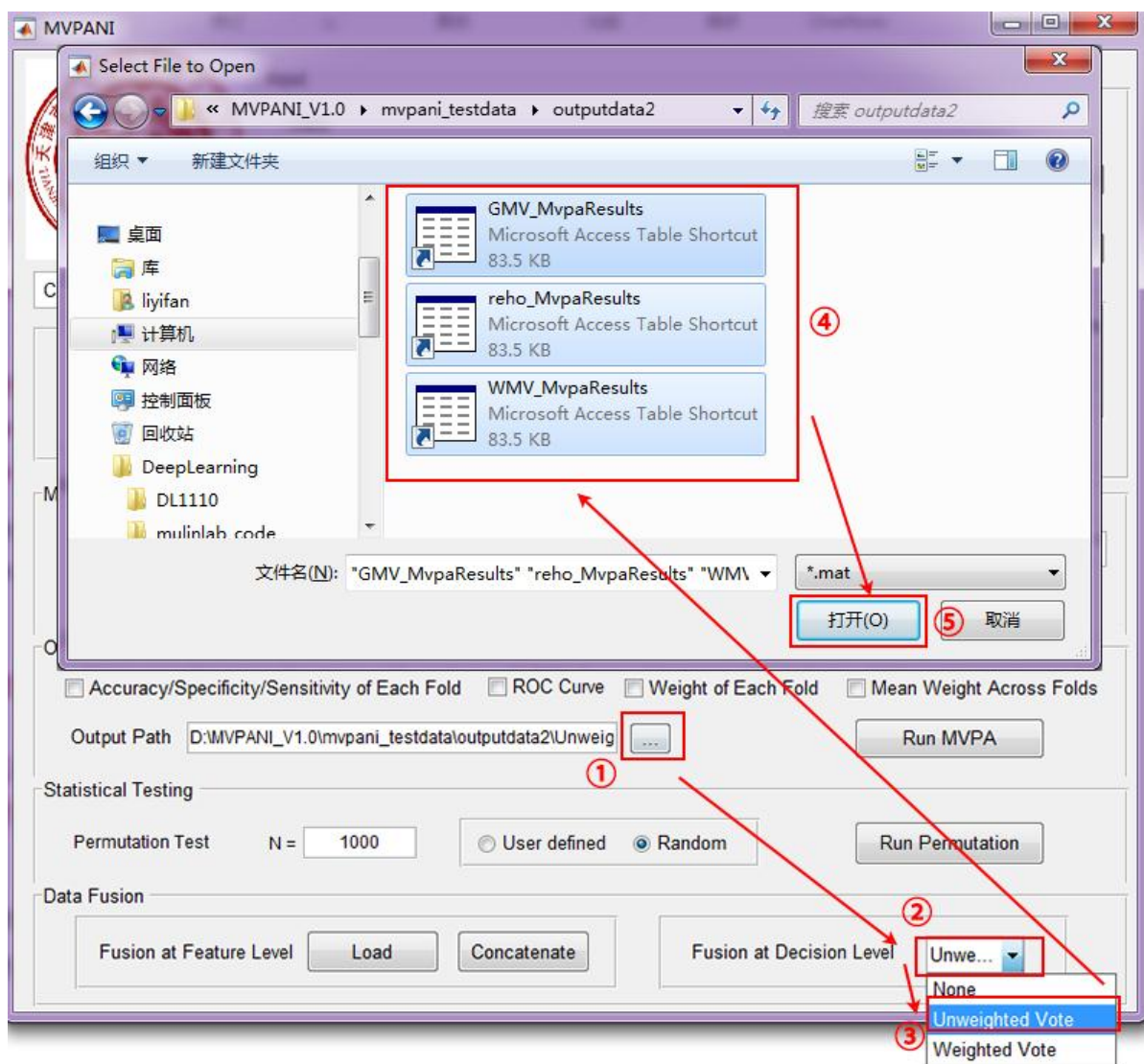


Figure 4.5.2.2-1 The operation steps for "Fusion at Decision Level" in Example 5. ① Click the '...' button to select the output path where the final results after fusion will be saved. ② Click the dropdown menu to select the method of decision fusion (e.g., Unweighted Vote). ③ Click "Unweight Vote" to open a window for loading the result files to be fused. ④ Select all result files (i.e., *_MvpaResults.mat files). ⑤ Click the "Open" button to generate the final result file after fusion.

UnweightedVoteResults	
1x1 struct with 5 fields	
Field ▲	Value
specificity_sensitivity	[0.6000,0.8000]
pred_label_vote	40x1 double
pred_label	40x3 double
true_label	40x1 double
Accuracy_vote	0.7000

Figure 4.5.2.2-2 Result of “Unweighted Vote”