SMAC (version 1.0) Manual

Hongtu Zhu @ BIG-S2

Leo Yu-Feng Liu @ BIG-S2

Jan. 11. 2017

With the development of imaging techniques, scientists are interested in identifying imaging biomarkers that are related to different subtypes or transitional stages of various cancers, neuropsychiatric diseases, and neurodegenerative diseases, among many others. We propose a novel Spatial Multi-category Angle-based Classifier (SMAC) for the efficient identification of such imaging biomarkers. The proposed SMAC not only utilizes the spatial structure of high-dimensional imaging data, but also handles both binary and multi-category classification problems. We introduce an efficient algorithm based on an alternative direction method of multipliers(ADMM) algorithm to solve the large-scale optimization problem for SMAC. Both our simulation and real data experiments demonstrate the usefulness of SMAC.

● Schematic overview of SMAC

The proposed SMAC is developed based on the Multi-category Angle based Classifiers (MAC), introduced by Zhang and Liu (2014) and the Large-margin Unified Machine (LUM), proposed by Liu et al. (2011). The LUM is basically a unified loss function which is characterized with two parameters, $a > 0$ and $c \geq 0$, in the following form:

$$l(u) = \begin{cases} 1-u, & if\ u \leq \frac{c}{1+c}, \\ \frac{1}{1+c}\left(\frac{a}{(1+c)u-c+a}\right)^a, & if\ u > \frac{c}{1+c}. \end{cases}$$

This loss function covers a lot of classifiers by varying the value of a and c. Please refer to the original paper for more information. In our package, we use a special case of the LUM loss, with $a \rightarrow +\infty$ and $c = 0$, which reduce the LUM to the following form.

$$l(u) = \begin{cases} 1-u, & if\ u \leq 0, \\ e^{-u}, & if\ u > 0. \end{cases}$$

This special loss is a hybrid of SVM and AdaBoost, which allows us to maximize the separation margin and dynamically assign weights in "weak" learners.

To handle the multi-category classification, we introduce the Multi-category Angle-based Classifier (MAC). The MAC basically maps the class labels from 1 to K ($K \geq 2$) into K simplex in the $K-1$ dimensional space, defined as following:

$$W_j = \begin{cases} (K-1)^{-\frac{1}{2}}\zeta, & if\ j = 1, \\ -\frac{1+K^{\frac{1}{2}}}{(K-1)^{\frac{3}{2}}}\zeta + \left(\frac{K}{K-1}\right)^{\frac{1}{2}} e_{j-1}, & if\ 2 \leq j \leq K, \end{cases}$$

where ζ is a vector with length $K-1$ and each element 1, and e_j is a vector in R_{k-1} such that its every element is 0, except the j-th element is 1. Given a set of data X in R^p, we are trying to use this classifier to find a function f: $R^p \rightarrow R^{K-1}$, and use the angle between f(X) and W_j to determine the predicted class label. See the following figure for an illustration.

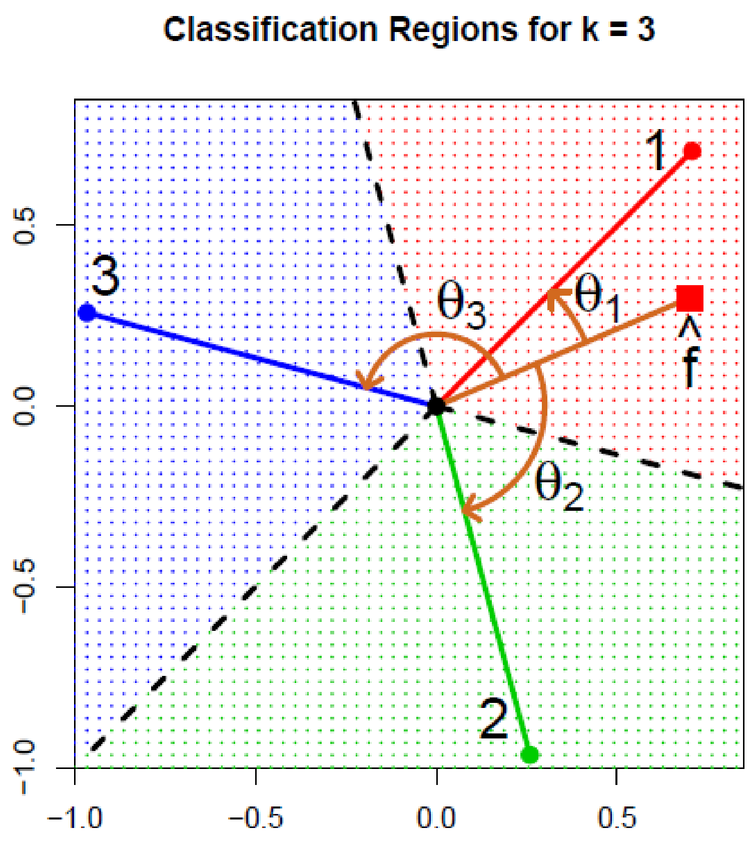


Fig. 1. Classification regions for $k = 3$. The mapped observation $\hat{f}$ is predicted as class 1 (red), because $\theta_1 < \theta_2 < \theta_3$. Observe that R^2 is naturally divided into three classification regions, plus the classification boundaries (dashed lines).

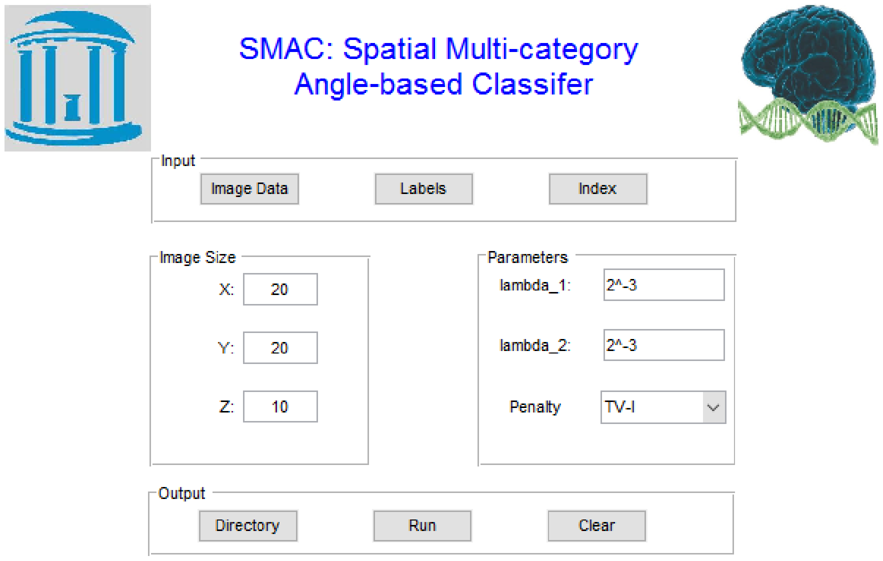
We introduce the generalized fused Lasso penalty (Tibshirani, 2011) to capture the spatial structure of imaging data. For the details about this penalty and the algorithm, we refer the users to the original paper of *SMAC: Spatial Multi-category Angle-based Classifier for High-dimensional Neuroimaging Data*.

● The GUI implementation

The GUI for the SMAC is implemented in MATLAB general user interface. It requires a MATLAB installed in your machine. After you download the package, please unzip the file into your desktop folder, for example

C:\Users\Administrator\Desktop\SMAC

Open the folder GUI and run SMAC_GUI.m in your MATLAB. This will pull up the windows for the SMAC_GUI (see following figure).



X: 20
Y: 20
Z: 10
lambda_1: 2^-3
lambda_2: 2^-3
Penalty TV-I

INPUT:

Image Data: click the ***Image Data*** button to load Image Covariates X. The image covariates should be save as a mat file which includes an n by p matrix, where n is the number of subjects and p is the dimension of the masked and vectorized images.

Labels: click the ***Labels*** button to select class labels for the n subjects. The labels should be save as a mat file which includes an n by 1 vector, where n is the number of subjects. The images should be labelled from 1 to K, where K is the total number of classes.

Index: click the ***index*** button to select index vector. The index vector should be save as a mat file which includes a p by 1 vector of element of 0 or 1, where 1 indicates the associated pixel/voxel to be included in the model. The size of the array should be matched with the size of the original image space, e.g. your image is a 3D image of size 20 by 20 by 10, then your index vector should be an 4000 dimensional vector.

IMAGE SIZE:

Specify your image size in the X Y Z boxes, where X, Y , Z are the total number of voxels along the 3 directions of your images. For 2D images, use Z = 1, and 1D curves use Y = Z = 1. Please be aware that your image size must match your index setting.

OPTIONS:

The SMAC includes 3 options for you to specify your models. Lambda 1 is the parameter for the sparsity penalizing level and Lambda 2 is for the spatial smoothness penalizing level. Both lambda 1 and 2 are nonnegative numbers. You can specify them in the form a number (e.g. 0.5) or an expression in MATLAB (e.g. 2^-5). Here we just provide a fixed parameter options in the GUI for a demonstration. If you want to tune the parameters, please refer to the later sections of the manual.

The popup menu for penalty type include the options of TV-I and TV-II, which corresponding to the first order total variation penalty and the second order total variation. Please refer to the original paper for a detail description of these two penalties.

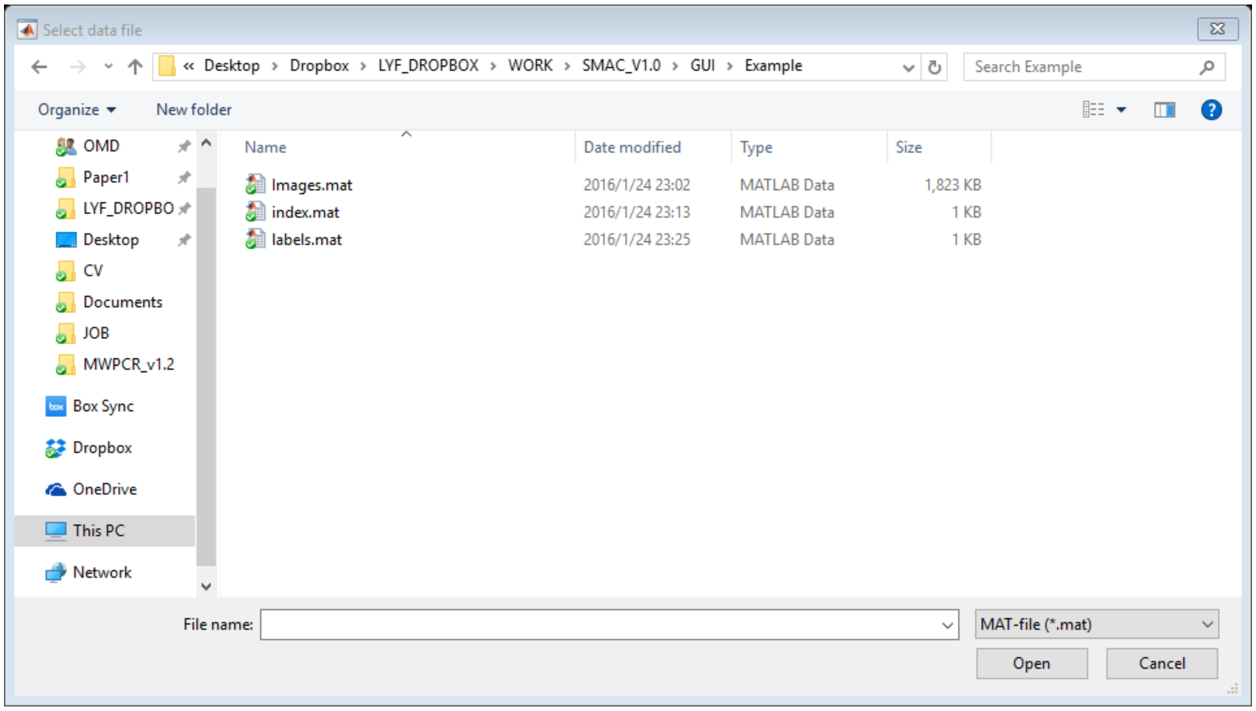
OUTPUT:

Output Directory: click the ***Output Directory*** button to select the directory for saving the results.

Run: click the ***Run*** button to start the algorithm for SMAC, after you have done all the previous steps.

Clear: click the ***Clear*** button to clear all the selected data and reset all the parameter selection.

- **GUI example:**



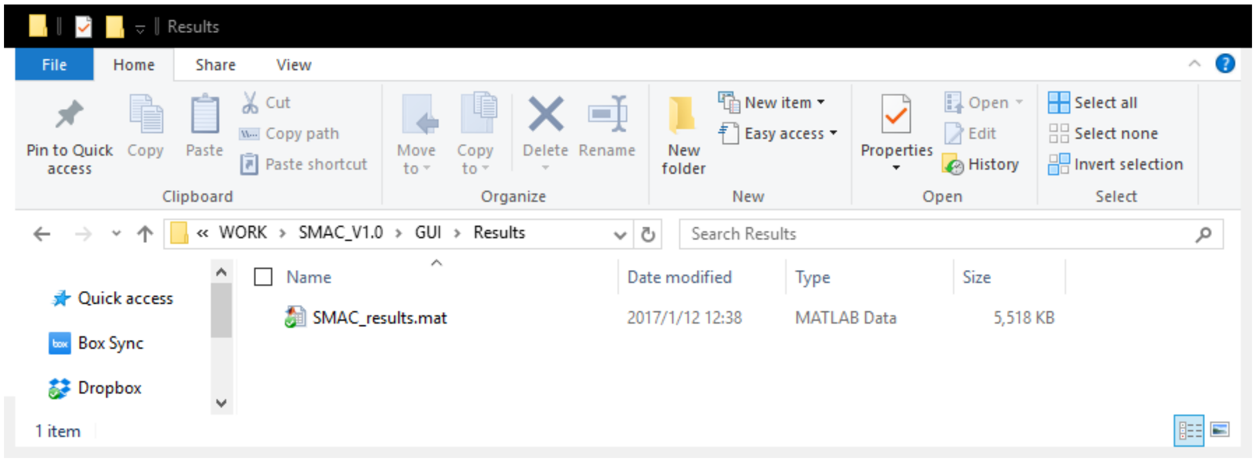
Name	Date modified	Type	Size
Images.mat	2016/1/24 23:02	MATLAB Data	1,823 KB
index.mat	2016/1/24 23:13	MATLAB Data	1 KB
labels.mat	2016/1/24 23:25	MATLAB Data	1 KB

We provide an example using the data included in the SMAC_v1.0.zip to demonstrate the use of the GUI. The detailed instruction is as follows:

1. Download SMAC_v1.0.zip and unzip on your local machine.
2. Open SMAC_GUI.m under the folder GUI in your MATLAB editor and hit F5 to run it.
3. Click the ***Image Data*** button, open the folder Example and select images.mat.
4. Click the ***Lables*** button, open the folder Example and select labels.mat.
5. Click the ***Index*** button, open the folder Example and select index.mat.
6. Set up the desired penalty parameter in lambda 1 and lambda 2 and choose your penalty type.

7. Click the ***Output Directory*** button and select Results folder, or any folder you want to save your output results.

8. Click the ***Run*** button to start the algorithm, and you will find the results is saved as SMAC_resutls.mat (see the figure below), which include the variables:



Name	Date modified	Type	Size
SMAC_results.mat	2017/1/12 12:38	MATLAB Data	5,518 KB

`'data'`: a MATLAB structure that including the data you used in the model;

`'out'`: a MATLAB structure that including the results;

Under the structure of `'out'`, you will find the following variables:

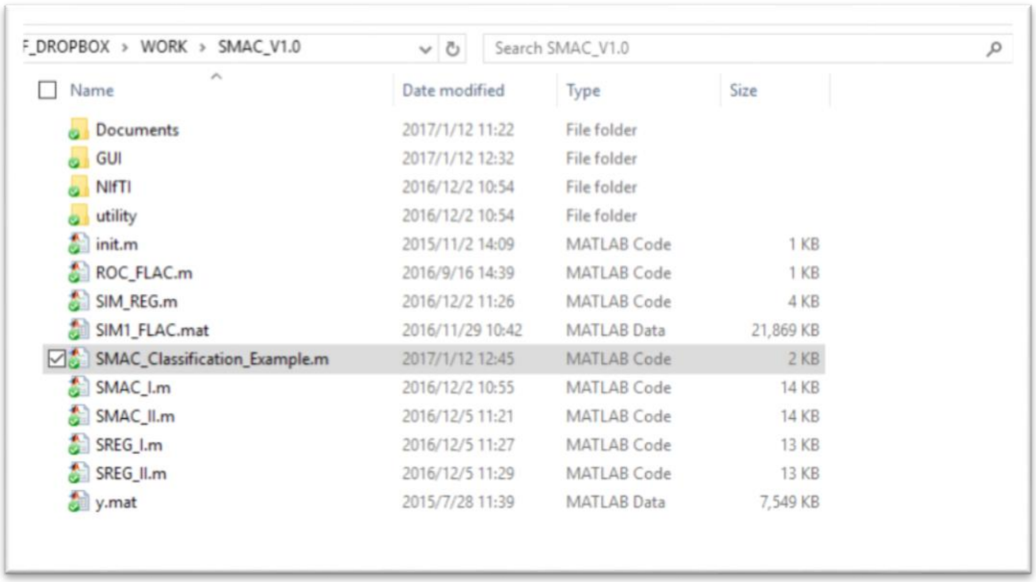
- `tune_acc`: the tuning accuracy
- `opt_lambda`: optimal lambda's chosen from the cross validation
- `b`: a vectorized coefficient image.
- `coef_1`: the coefficient image.
- `pred_y`: predicted class labels.

- `test_acc`: testing accuracy evaluated on the testing set.
- `conf`: confusion matrix of the testing results.

● The script based implementation with cross validation and ROC analysis

The GUI implementation is a simple demonstration of the package, many functions of our package cannot be achieved through the GUI. Here we introduce a script based pipeline, so that you can use for a larger job, and conduct the cross validation and ROC analysis on your classification problems.

To run such jobs, please follow the script named "SMAC_Classification_Example.m" under the folder SMAC (see the following figure).



F_DROPBOX › WORK › SMAC_V1.0 — Search SMAC_V1.0

Name	Date modified	Type	Size
Documents	2017/1/12 11:22	File folder	
GUI	2017/1/12 12:32	File folder	
NIfTI	2016/12/2 10:54	File folder	
utility	2016/12/2 10:54	File folder	
init.m	2015/11/2 14:09	MATLAB Code	1 KB
ROC_FLAC.m	2016/9/16 14:39	MATLAB Code	1 KB
SIM_REG.m	2016/12/2 11:26	MATLAB Code	4 KB
SIM1_FLAC.mat	2016/11/29 10:42	MATLAB Data	21,869 KB
SMAC_Classification_Example.m	2017/1/12 12:45	MATLAB Code	2 KB
SMAC_I.m	2016/12/2 10:55	MATLAB Code	14 KB
SMAC_II.m	2016/12/5 11:21	MATLAB Code	14 KB
SREG_I.m	2016/12/5 11:27	MATLAB Code	13 KB
SREG_II.m	2016/12/5 11:29	MATLAB Code	13 KB
y.mat	2015/7/28 11:39	MATLAB Data	7,549 KB

This is the classification simulation in the original paper, for more information about the setup of the example, please refer to the paper.

Please follow the pipeline below to set up an analysis in this example:

1. Load your images and create the mask accordingly.

2. Masking out the irrelevant region and save the masked image covariates as a n by p matrix. Here p is the total dimension after masking. Save the index as the vectorized mask matrix.

3. Split your data into training, tuning and testing (Optional).

4. Change the data file name and path accordingly in the following part:

```
%% Load the image data file.
load SIM1_FLAC.mat

%% Assign the class label accordingly.
Y = [ones(30,1);ones(30,1)*2]; % Class labels must be from 1 to K.
```

5. Assign the training, tuning and testing data accordingly (Optional), if you just want to do a training model, please set the three data sets to be identical.

```
%% Assign the training, tuning and testing data set.
data.train_x = SIM1.train; % train images, n by p
data.train_y = Y; % train labels, n by 1
data.tune_x = SIM1.tune; % tune images, n by p
data.tune_y = Y; % tune labels, n by 1
data.test_x = SIM1.test; % test images, n by p
data.test_y =  [ones(300,1);ones(300,1)*2]; % test labels, n by 1
```

6. Assign the index vector and image size according to your image data and masking matrix.

```
%% Assign the index vector.
```

```
% You can simply vectorize your mask matrix to get this. For a full image
% without masking, please use all ones, or skip this step, the program will
% give a default value for this.
data.index = ones(4000,1);

%% Assign the image size.
imagesize = [20,20,10];
```

7. Assign the parameter to specify your model. You can also skip this step, the system will choose the default values to build your model.

```
%% Assign the options for the algorithm.
% You can skip this step; all the options will have a default values.
options.itmax=1000;  % Max iteration, default is 500.
options.rho=1; % Lagrangian parameter, default is 1.
options.l1flag =0; % Indicator of penalizing intercept. Default is 0.
options.tol=5E-4; % Tolerance for the convergence of the algorithm. Default is 10E-4.
options.progress = false; % Display the tuning progress. Default is false.
```

Note: the following 2 parameters are very important, since they determine the range for you tuning parameter.

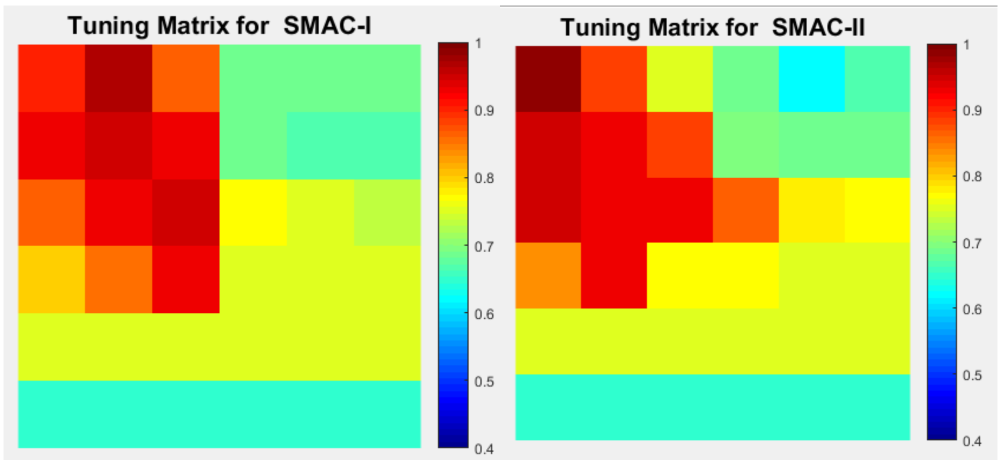
```
options.lambda1 = 2.^[-5:2:5]; % Penalty level for L1. Default is 8.^(-3:3);
options.lambda2 = 2.^[-5:2:5]; % Penalty level for TV. Default is 8.^(-3:3);
```

8. Run the following part without editing anything. It will automatically train your model using the tuning parameters you provided and select the best model based on the tuning accuracies.

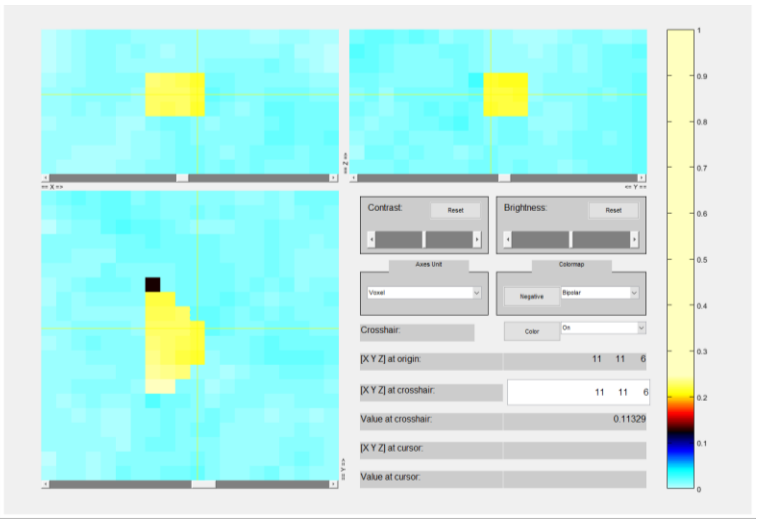
```
%=====================Do Not Edit======================%
% Do not edit the following part, unless you really understand the code.
%% The beginning of the program
run('init.m') % add the full path of the package
```

```
[out1] = SMAC_I (data,imagesize,options); % run the classification with TV-I penalty

[out2] = SMAC_II (data,imagesize,options); % run the classification with TV-II penalty

%% Plot the tuning accuracy
figure; Tuning_Matrix_Plot(out1.tune_acc, 'SMAC-I') % Plot the tuning accuracy
figure; Tuning_Matrix_Plot(out2.tune_acc, 'SMAC-II')

%% Plot the coefficient images for the optimal model
k = length(unique(data.train_y));
for iii = 1:k-1
    eval( sprintf('view_nii(make_nii(-out1.coef_%d));',iii));
    colormap(fireice);
end

for iii = 1:k-1
    eval( sprintf('view_nii(make_nii(-out2.coef_%d));',iii));
    colormap(fireice);
end
 %% Plot the ROC curves and compute the AUC
[AUC1] = ROC_analysis(data.test_y,data.test_x*out1.coef_1(:),1,'SMAC-I');
[AUC2] = ROC_analysis(data.test_y,data.test_x*out2.coef_1(:),1,'SMAC-II');
  %==================================================%
```

9. The tuning matrices will be plotted in the following figures.



10.The estimated coefficient images from the optimal model will be displayed.



Voxel

Bipolar

On

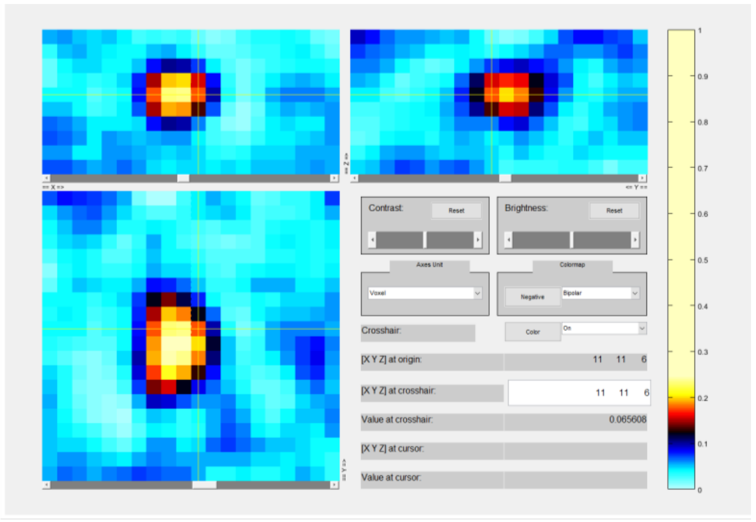
[X Y Z] at origin: 11 11 6

[X Y Z] at crosshair: 11 11 6

Value at crosshair: 0.11329

[X Y Z] at cursor:

Value at cursor:



Voxel

Bipolar

On

[X Y Z] at origin: 11 11 6

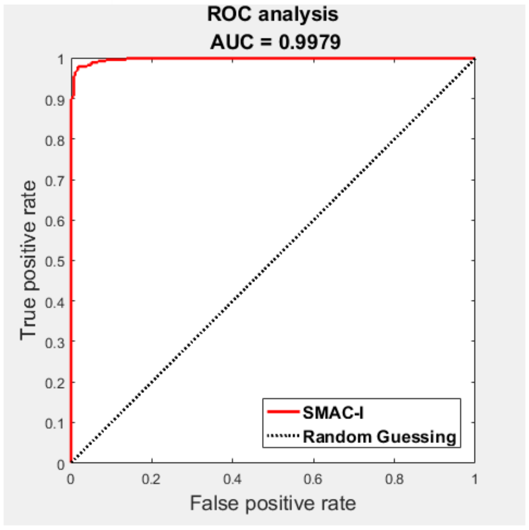
[X Y Z] at crosshair: 11 11 6

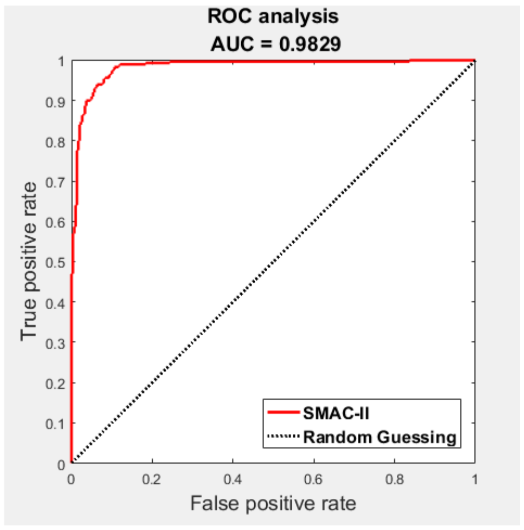
Value at crosshair: 0.065608

[X Y Z] at cursor:

Value at cursor:

11. The ROC analysis will be conducted with the ROC curves plotted in the last step.





If you have any questions about this package, please contact Leo Yu-Feng Liu via yfliu86@live.unc.edu.

● Reference:

Leo Yu-Feng Liu, Yufeng Liu and Hongtu Zhu. “SMAC: Spatial Multi-category Angle-based Classifiers for Neuroimaging Data.” *NeuroImage,* submitted.

Zhang, Chong, and Yufeng Liu. "Multicategory angle-based large-margin Classification." *Biometrika* 101.3 (2014): 625-640.

Liu, Yufeng, Hao Helen Zhang, and Yichao Wu. "Hard or soft classification? Large-margin unified machines." *Journal of the American Statistical Association* 106.493 (2011): 166-177.