MWPCR (version 1.2) Manual

Hongtu Zhu @ BIG-S2

Leo Yu-Feng Liu @ BIG-S2

Jan. 1. 2017

The multiscale weighted principal component regression (MWPCR) framework is used to make use of high dimensional features with strong spatial features (e.g., smoothness and correlation) to predict an outcome variable, such as disease status. This development is motivated by identifying imaging biomarkers that could potentially aid detection, diagnosis, assessment of prognosis, prediction of response to treatment, and monitoring of disease status, among many others. The MWPCR can be regarded as a novel integration of principal components analysis (PCA), kernel methods, and regression models. In MWPCR, we introduce various weight matrices to pre-whiten high dimensional feature vectors, perform matrix decomposition for both dimension reduction and feature extraction, and build a prediction model by using the extracted features. Examples of such weight matrices include an importance score weight matrix for the selection of individual features at each location and a spatial weight matrix for the incorporation of the spatial pattern of feature vectors. Our MWPCR can be a useful statistical tool that can integrate the importance score weights with the spatial weights and recover the low dimensional structure of high dimensional features.

Schematic overview of MWPCR

The proposed MWPCR consists of two components: a low-rank model for multi-scale weighted PCA (MWPCA) and a prediction model.

Let Q be a p by p weight matrix. The low-rank model for MWPCA can be written as

$$\tilde{X} = (X - 1_n\mu^T)Q = UDV^T + \epsilon = \sum_{k=1}^{K} u_k d_k v_k^T + \epsilon$$

where U, D and V are respectively, n by K, K by K, and p by K matrices such that diag(D) ≥ 0 and $U^TU = V^TV = I_K$, where I_K is a K by K identity matrix.

Once we have these principle score, we then build a prediction model $\mathrm{R}(\mathrm{y_i}; \mathrm{U_i}, \theta)$ with $\mathrm{y_i}$ as response and $\mathrm{U_i}$ as covariates, where θ is a vector of unknown (finite-dimensional or non-parametric) parameters.

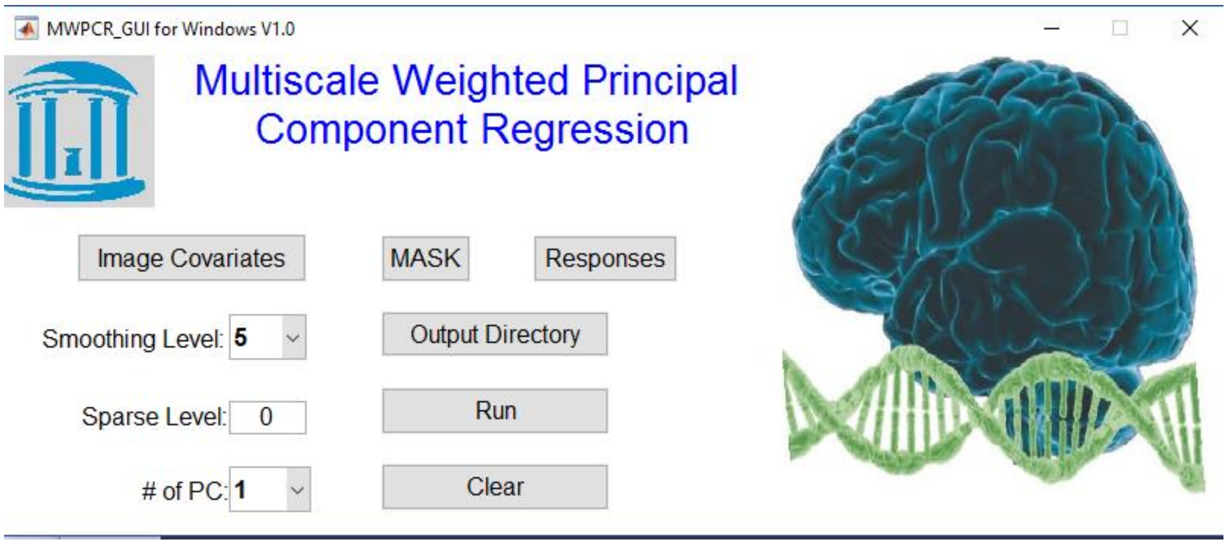
For more details about MWPCR, please refer to the original paper of *MWPCR: Multiscale Weighted Principal Component Regression for High-dimensional Prediction* (to be published in JASA).

The GUI implementation

The GUI for the Multiscale Weighted Principal Component Regress is implemented in Matlab general user interface. It require a Matlab installed in your local machine. After you download the package, please unzip the file into your desktop folder, for example

C:\Users\Administrator\Desktop\MWPCR

Open the folder GUI and run MWPCR_GUI_V1.m in your Matlab. This will pull up the windows for the MWPCR_GUI (see following figure).



INPUT:

Image Covariates: click the ***Image Covariates*** button to select 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.

Responses: click the ***Response*** button to select response variable Y. The responses should be save as a mat file which includes an n by 1 vector, where n is the number of subjects. It can be discrete or continuous, depending on the associated problems.

MASK: click the ***MASK*** button to select masking matrix M. The masking matrix should be save as a mat file which includes a 2D/3D array 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 160 by 160 by 96, then your mask matrix should be an 160 by 160 by 96 array.

OPTIONS:

Smoothing Level: click the popup menu ***Smoothing Level*** to select the smoothing level (S) from 0 to 5. 0 means no smoothing in the MARM algorithm. 5 means max smoothing in the MARM algorithm.

Sparse Level: enter the threshold (t) in edit blank ***Sparse Level*** for creating the global importance weight matrix. The threshold should be a number between 0 and 1. 0 means no thresholding at all, i.e. all voxels are included in the analysis. 1 means no voxels will be included (not recommended).

# of PC: click the popup menu (***# of PC***) to select the number of principle components (K) from 1 to 20.

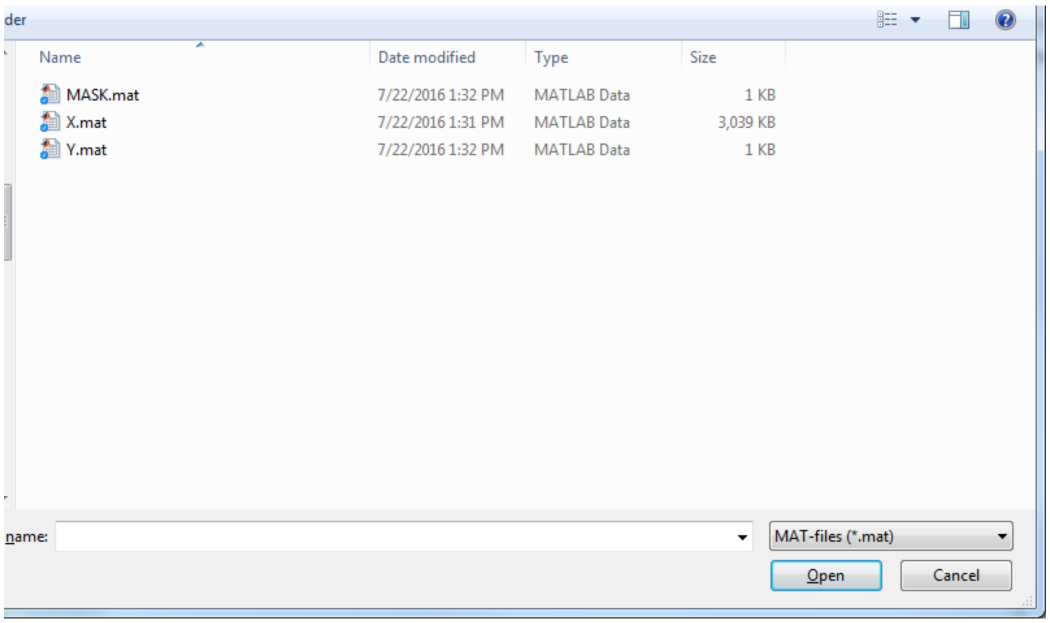
OUTPUT:

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 MWPCR, 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
MASK.mat	7/22/2016 1:32 PM	MATLAB Data	1 KB
X.mat	7/22/2016 1:31 PM	MATLAB Data	3,039 KB
Y.mat	7/22/2016 1:32 PM	MATLAB Data	1 KB

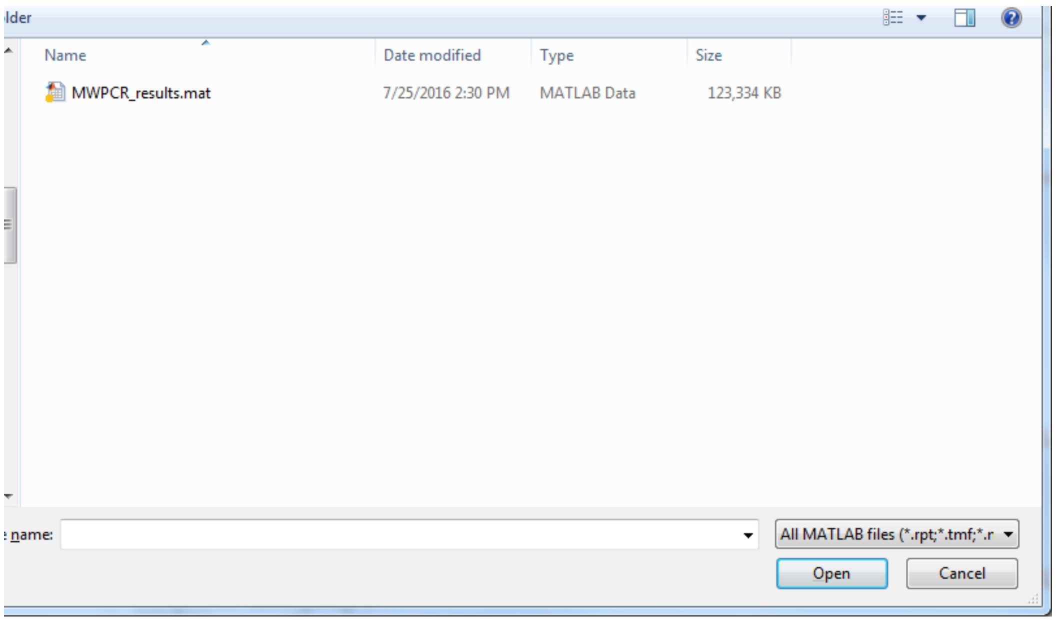
We provide an example using the data included in the MWPRR.zip to demonstrate the use of the GUI. The detailed instruction is as follows:

1. Download MWPCR.zip and unzip on your local machine.

2. Open MWPCR_GUI_V1.m in your Matlab editor and hit F5 to run it.

3. Click the ***Image Covariates*** button, open the folder Example and select X.mat.
4. Click the ***Response*** button, open the folder Example and select Y.mat.
5. Click the ***MASK*** button, open the folder Example and select MASK.mat.
6. Set up the desired smooth level, sparse level, and number of PC’s.
7. Click the ***Output Directory*** button and select Results folder.
8. Click the ***Run*** button to start the algorithm, and you will find the results is saved as MWPCR_resutls.mat, which include the variables:

```
'X': the imaging covariates;

'Y': the responses;

'MASK': the masking matrix;

'mask_h': the header file for MASK;

'Q': the weighting matrix;

'U': the rows spaces for PCA.

'D': the eigen values;

'V': the PC loadings;

'Beta1': the regression coefficient.
```



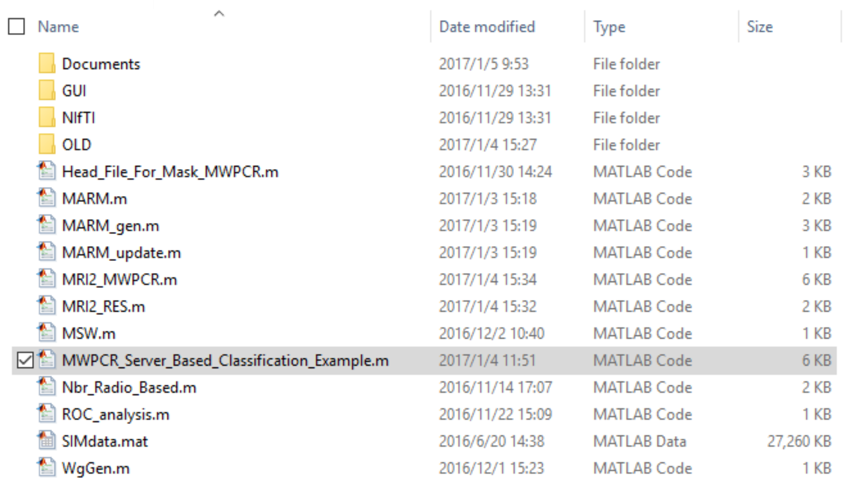
MWPCR_results.mat 7/25/2016 2:30 PM MATLAB Data 123,334 KB

All MATLAB files (*.rpt;*.tmf;*.r

Then you can use X_new = X*Q as the new covariate matrix to do either PCA or any other analysis.

● The server based implementation with cross validation

The server based implementation is designed for a large-scale job that requires more running time and memory. To run such jobs, please follow the script named "MWPCR_Server_Based_Classification_Example.m" under the folder MWPCR (see the following figure). This is the classification simulation in the paper, for more information about the setup of the example, please refer to the original paper, Zhu, Hongtu, Dan Shen, Xuewei Peng, and Leo Yufeng Liu. "MWPCR: Multiscale Weighted Principal Component Regression for High-dimensional Prediction." *Journal of the American Statistical Association* in press, 2016.



Name	Date modified	Type	Size
Documents	2017/1/5 9:53	File folder	
GUI	2016/11/29 13:31	File folder	
NIfTI	2016/11/29 13:31	File folder	
OLD	2017/1/4 15:27	File folder	
Head_File_For_Mask_MWPCR.m	2016/11/30 14:24	MATLAB Code	3 KB
MARM.m	2017/1/3 15:18	MATLAB Code	2 KB
MARM_gen.m	2017/1/3 15:19	MATLAB Code	3 KB
MARM_update.m	2017/1/3 15:19	MATLAB Code	1 KB
MRI2_MWPCR.m	2017/1/4 15:34	MATLAB Code	6 KB
MRI2_RES.m	2017/1/4 15:32	MATLAB Code	2 KB
MSW.m	2016/12/2 10:40	MATLAB Code	1 KB
MWPCR_Server_Based_Classification_Example.m	2017/1/4 11:51	MATLAB Code	6 KB
Nbr_Radio_Based.m	2016/11/14 17:07	MATLAB Code	2 KB
ROC_analysis.m	2016/11/22 15:09	MATLAB Code	1 KB
SIMdata.mat	2016/6/20 14:38	MATLAB Data	27,260 KB
WgGen.m	2016/12/1 15:23	MATLAB Code	1 KB

The following is a pipeline 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.

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

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

```
%% Loading the data
% Please modify the data file path and name accordingly for your own job.
load ('SIMdata.mat') % This is the simulation data from the paper;

% Please use the mask file according your own data.
mask = ones(20,20,10);
imagesize = size(mask); % the image size must match the mask size.
```

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.

```
%% Load the data, this is an example for 3-way split data cross validation
data = SIM1;
% load the training data;
X_tr = data.train;
Y_tr = [ones(60,1);zeros(40,1)];
% Load the tuning data;
X_tune = data.tune;
% Center the tuning data;
X_tune = X_tune-repmat(mean(X_tune,1),size(X_tune,1),1);
Y_tune = [ones(60,1);zeros(40,1)];
% Load the testing data;
X_test = data.test;
% Center the testing data;
X_test = X_test-repmat(mean(X_test,1),size(X_test,1),1);
Y_test = [ones(60,1);zeros(40,1)];
```

Note this is a classification example, and we are using a weighted threshold in order to get a better performance:

```
[n,p] = size(X_tr);
% Compute the threshold for classification
prob_thres = sum(Y_tr)/n;
```

6. Assigning the tuning range for the cross validation. In this example, we are tuning for the sparsity level and the number of principle components.

```
%% 2D penal cross validation on a classification analysis
% In this example, we tune the model by selecting different sparsity level
% and the number of PC components.

Sparsity_tune_n = 21; % number of sparsity level being tuned.
PC_num_tune_n = 21; % number of PC's being tuned.

Thres_LB = 0.9; % the lower bound of sparsity level, between 0 and 1;
Thres_UB = 0.99; % the upper bound of sparsity level, between 0 and 1;
thres_pool = linspace(Thres_LB,Thres_UB,Sparsity_tune_n);
```

7. Run the following part without editing anything. It generates the neighborhood information of the images, computes the adaptive multi-scale coefficients, and create the weight matrix Q in the paper. After these steps, a cross validation is conducted to selected the optimal parameters and build the final model.

```
%=====================Do Not Edit=====================%
% Assign the parameters according to your image types.
% h is the base searching radius, we recommend h = 1.5 for 1D; 1.28 for 2D; 1.2 for 3D;
% S is the max power of radius, we recommend S = 3 for 1D; 4 for 2D; 5 for 3D;

dim = length(imagesize);
switch dim
```

```
    case 1
        h = 1.5; S = 3;
    case 2
        h = 1.28; S = 4;
    case 3
        h = 1.2; S = 5;
end
%% Compute the head file, it contains all the neighborhood information.
[Mask_Idx, Mask_Loc,Nbr_Dist_2] = Head_File_For_Mask_MWPCR(mask,h^S);

%% Compute the multi-scale adpative coefficients.
[beta,beta_cov,WE]  = MARM (Mask_Idx,Nbr_Dist_2,X_tr,Y_tr,h,S);

%% Begin cross validation analysis
Tune_ACC = zeros(Sparsity_tune_n,PC_num_tune_n);
Tune_AUC = zeros(Sparsity_tune_n,PC_num_tune_n);

% outter loop is for tuning the sparsity level
for j =1:Sparsity_tune_n
    thres = thres_pool(j); % this is a threshold parameter, comtrol the sparseness of the estimation.
    [Q,~] = MSW (WE,beta,beta_cov,thres);
    %% Project the data using the weight matrix Q.
    X_t = (X_tr-repmat(mean(X_tr,1),n,1))*Q; % X_t is X_tilde in the paper

    %% PCA analysis on the projected data
    [U,D,V] =   svd(X_t, 'econ');
    PC_score = U*D; % PC scores;
    PC_loading = V; % PC loadings;
% inner loop is for tuning the number of pricinple components
for i = 1:PC_num_tune_n
        PC_num = i;
        % Create the design matrix with the PC scores;
        X_mwpcr = [ones(n,1),PC_score(:,1:PC_num)];
        % Calculate the regression coefficients;
        Beta_cla = (X_mwpcr'*X_mwpcr)^-1*X_mwpcr'*Y_tr;
        % Project the tuning data using the weight matrix and the PC direction
        X_mwpcr_tune = X_tune*Q*PC_loading;
        % Create the design matrix for tuning data;
        X_mwpcr_tune = [ones(size(X_tune,1),1),X_mwpcr_tune(:,1:PC_num)];
        % Compute the classification scores for the tuning data;
        Y_hat_tune = X_mwpcr_tune*Beta_cla;
        % Threshold the scores to get the class labels;
        Y_tune_predict = (Y_hat_tune>prob_thres);
        % Compute the classification accuracy;
        Tune_ACC(i,j) = sum(Y_tune_predict==Y_tune)/length(Y_tune);
        [~,~,~,Tune_AUC(i,j)] = perfcurve(Y_tune, Y_hat_tune,1);
    end
end

%% Tuning parameter selection procedure
% First criteria: based on tuning accuracy
max_index = find(Tune_ACC==max(Tune_ACC(:)));
% If tied, use second criteria: based on tuning AUC
if length(max_index)>1
    max_index_index = find(Tune_AUC(max_index)==max(Tune_AUC(max_index)));
    max_index = max_index(max_index_index);
end
[PC_num_max, thres_max] = ind2sub([Sparsity_tune_n,PC_num_tune_n],max_index);
% if tied, use third criteria: fewer principle components and higher
% sparsity level
PC_num_index = find(PC_num_max == min(PC_num_max));
PC_num_max = min(PC_num_max);
```

```
thres_max = max(thres_max(PC_num_index));

%% Compute the testing results, based on the selected model
thres = thres_pool(thres_max);
PC_num = PC_num_max;
% Retrain the model using the slected parameter
[Q,Wg] = MSW (WE,beta,beta_cov,thres);
X_t = (X_tr-repmat(mean(X_tr,1),n,1))*Q;
[U,D,V] =   svd(X_t, 'econ');
PC_score = U*D;
PC_loading = V;
X_mwpcr = [ones(n,1),PC_score(:,1:PC_num)];
Beta_cla = (X_mwpcr'*X_mwpcr)^-1*X_mwpcr'*Y_tr;
% Final model is retrained and ready for prediction

% Project the test using the weight matrix and the PC direction
X_mwpcr_test = X_test*Q*PC_loading;
% Create the design matrix for testing data;
X_mwpcr_test = [ones(size(X_test,1),1),X_mwpcr_test(:,1:PC_num)];
% Compute the classification scores for the testing data;
Y_hat_test = X_mwpcr_test*Beta_cla;
% Threshold the scores to get the class labels;
Y_test_predicted = (Y_hat_test>prob_thres);
% Compute the classification accuracy;
Test_ACC = sum(Y_test_predicted==Y_test)/length(Y_test);

MWPCR_coef = reshape(Q*PC_loading(:,1:PC_num_max)*Beta_cla(2:end),imagesize);
MWPCR_tune = Tune_ACC;
MWPCR_test = Test_ACC;
MWPCR_AUC = ROC_analysis(Y_test, Y_hat_test,1,'MWPCR'); % display the ROC curves
nii_view(MWPCR_coef) % display the estimated coefficient image
```

```
Test_ACC % display the testing accuracy
%=====================================================%
```

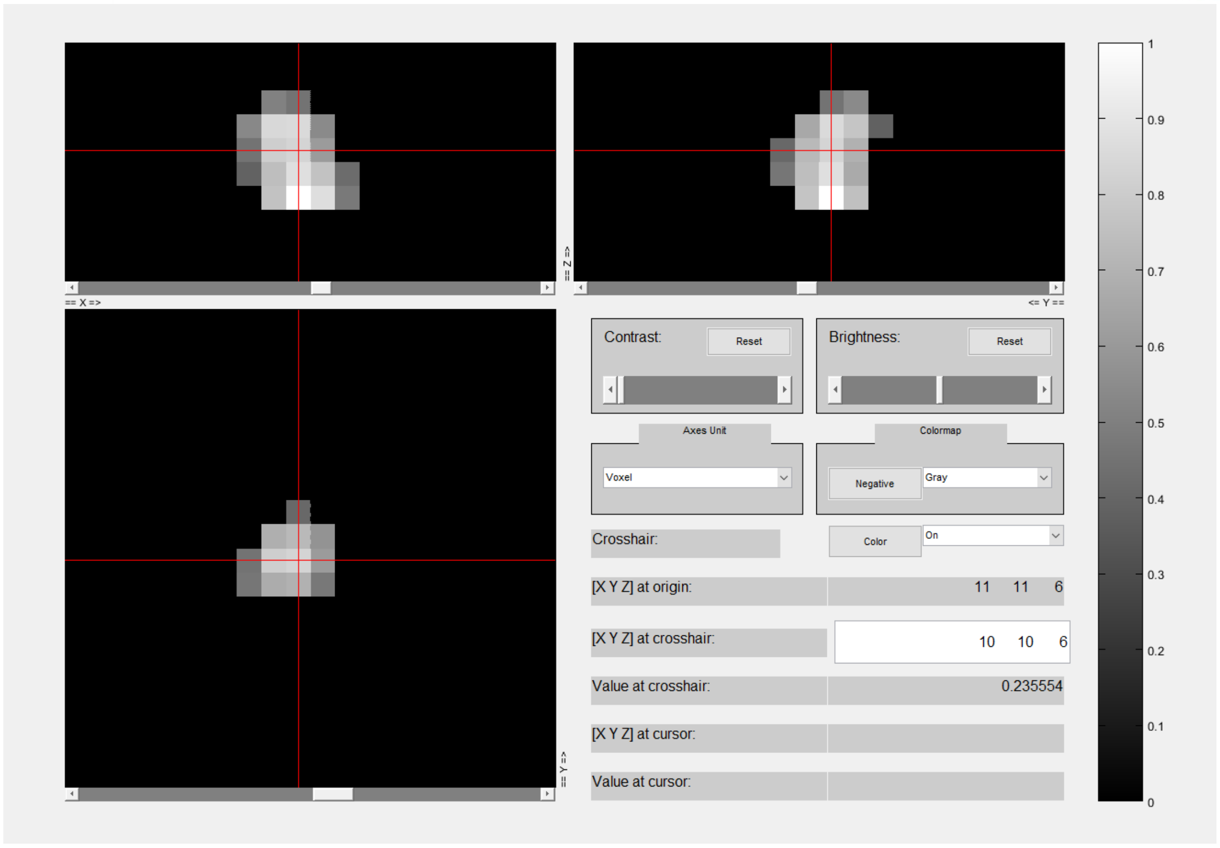
8. The progress of the program is displayed in form of the percentage of the finished job. The final testing accuracy is displayed at the end of the procedure.

```
+++++++Cross validation procedure started!!+++++++
+++++++0.23% of the job is done!!+++++++
+++++++0.45% of the job is done!!+++++++
……
+++++++99.77% of the job is done!!+++++++
+++++++100.00% of the job is done!!+++++++
+++++++Cross validation procedure is finished!!+++++++
+++++++The final testing accuracy is 0.92+++++++
```

9. Run the following part to plot the global important scores.

```
%% Log_10 P-value plot
% For 1 D images
% plot(Wg);
% For 2D/3D images
Wg = reshape(Wg,imagesize);
view_nii(make_nii(Wg)); % This is a 3-D view of the important score matrix.
```

This will generate the following figure, indicating the global important regions in the images (brighter color means more important).



Contrast:

Brightness:

Voxel

Gray

Crosshair: On

[X Y Z] at origin: 11 11 6

[X Y Z] at crosshair: 10 10 6

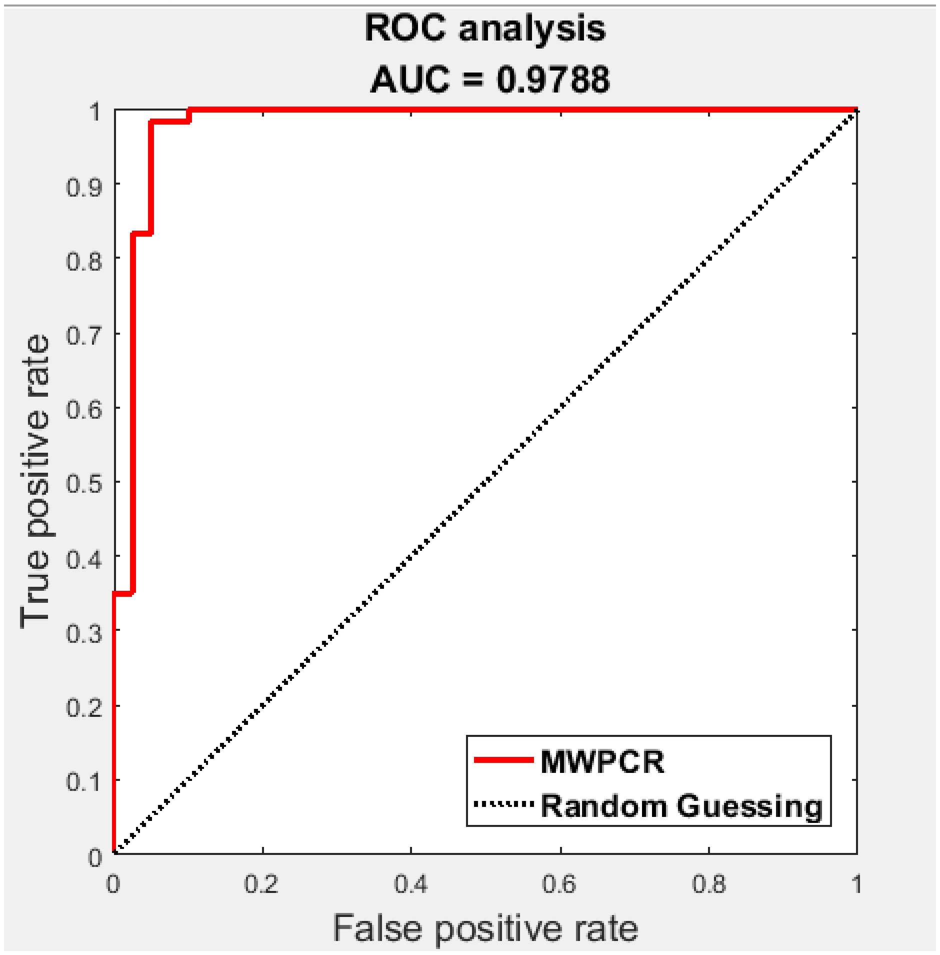
Value at crosshair: 0.235554

[X Y Z] at cursor:

Value at cursor:

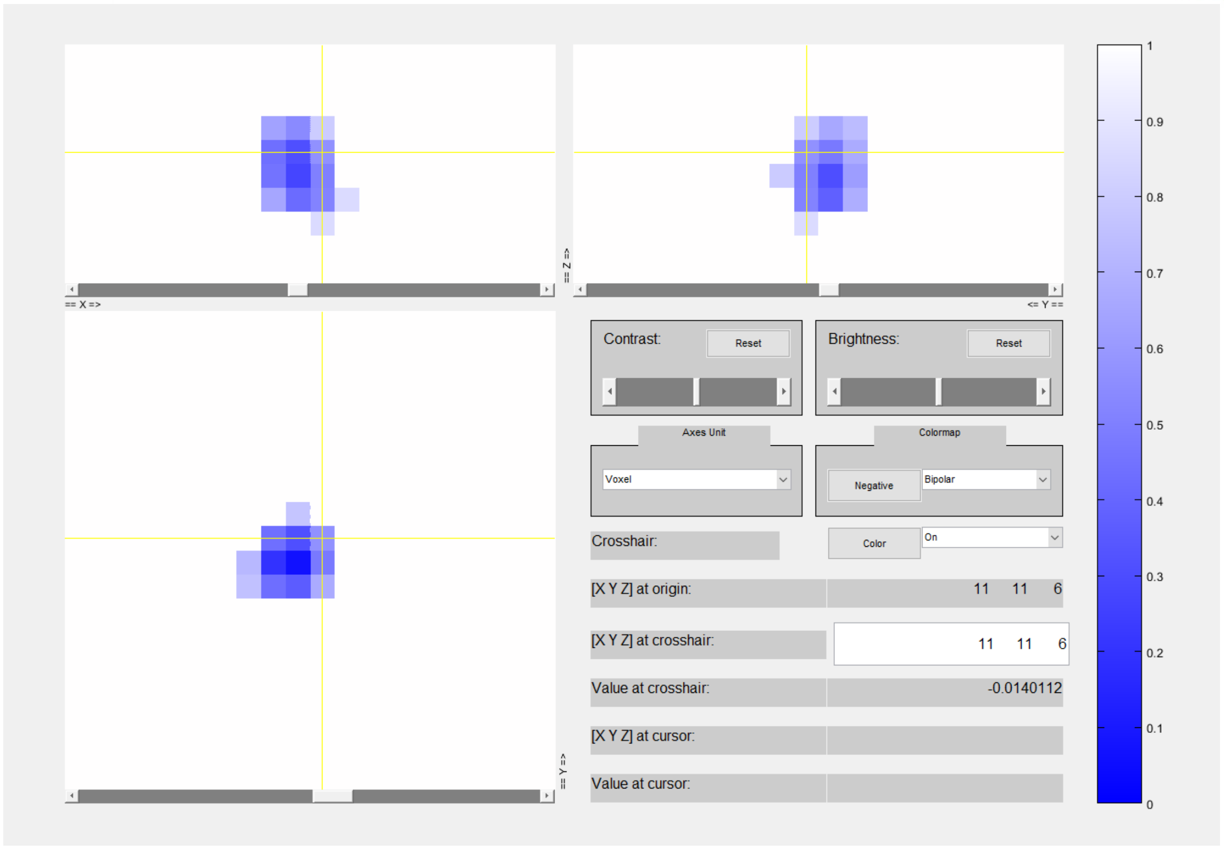
10. Conduct the ROC analysis and generate the ROC curves.

```
%% ROC curves plot
MWPCR_AUC = ROC_analysis(Y_test, Y_hat_test,1,'MWPCR'); % display the ROC curves
```



11.Display the estimated coefficient images:

```
%% Display the estimated coefficient image
nii_view(MWPCR_coef)
```



Contrast:

Brightness:

Axes Unit: Voxel

Colormap: Negative Bipolar

Crosshair: On

[X Y Z] at origin: 11 11 6

[X Y Z] at crosshair: 11 11 6

Value at crosshair: -0.0140112

[X Y Z] at cursor:

Value at cursor:

● Server based real data example: ADNI MRI classification (AD v.s. NC)

For real medical image analysis problem, we recommend a parallel computing for our package. Here, we have included an example to illustrate the pipeline for dealing with large scale imaging data with cross validation implemented. This example is about the Alzheimer's Disease Neuroimaging Initiative (ADNI) MRI data classification. For more information about the data set, please refer to http://adni.loni.usc.edu/.

In our example, we did a ten-fold cross validation, with an inner nine-fold cross validation to select the tuning parameters. We tuned the number of principle components and the sparsity levels simultaneously from an 11 by 11 2D grid search. Thus the procedure includes running the MWPCR for 10890 (10*9*11*11=10890) times.

After finish the 10890 runs of MWPCR, we have provided a program to collect and summarize the results. This code will select the best model based on the following three criteria:

1. Select the model(s) with the best tuning classification accuracy;
2. If we have multiple models with the best tuning classification accuracy, we select the one(s) with the best tuning classification AUC among them;
3. If we have multiple models after step 2, we select the one with highest sparsity level with the least number of principle components.

The final testing results will be an average based on each of the ten folds, and this will be a very comprehensive evaluation of the model based on the 10890 training models.

The example bash command of submitting such code is provided in the text file named “batch_submit_for_MWPCR.txt”. The source code for running the MWPCR is included in the script named “MRI2_MWPCR.m” and the source code for collecting the results is in the script “MRI2_RES.m”. For your own job, you may need to refer to the source codes in the two Matlab scripts. The scripts are well annotated, so that they can be easily modified to accommodate your own job.

The data in this package is for illustration purpose only. Please do not use them for any other purposes without the permission of the author.

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

Reference:

Zhu, Hongtu, Dan Shen, Xuewei Peng, and Leo Yufeng Liu. "MWPCR: Multiscale Weighted Principal Component Regression for High-dimensional Prediction." *Journal of the American Statistical Association* in press, 2016. .