Functional Network Connectivity and SVM Classification of fMRI data using **R**

Ariana Anderson, Ph.D., Mark S. Cohen, Ph.D.
UCLA Center for Cognitive Neuroscience
760 Westwood Plaza, Suite 17-369
Los Angeles, CA 90095
E-mail: ariana82 @ ucla . edu

March 15, 2011

Abstract

We demonstrate and provide **R** code that can classify between groups of fMRI scans based on functional network connectivity differences using the packages **analyzefMRI**, **vegan**, **igraph**, and **e1071**. We run ICA on fMRI data to establish functional networks, measure the functional connectivity between these networks using the temporal crosscorrelations between independent component, and then transform the connectivities of each scan into a graph structure. Connectivity properties of these graph structures are extracted, and used as a feature matrix for an SVM classifier. Collectively, this manuscript provides code that requires the user only to supply a list of filenames to be processed, the fMRI scans in Analyze or NIFTI format, and the scan labels, requiring only 4 lines of code to be altered. All other parameters are inferred from the data to make these routines adaptable for general usage. This code can also be altered to perform connectivity analysis and classification using ROI based methods by reading in distance arrays previously created. Collectively, this paper provides and explains both methods and code to perform functional network connectivity and fMRI SVM classification. The code is freely available on the NITRC website at `http://www.nitrc.org/projects/fmriclassify/`.

1 Introduction

Functional Magnetic Resonance Imaging (fMRI) is a four-dimensional medical imaging modality that captures changes in blood oxygenation over time, an indirect measure of neuronal activation. An increasing focus of interest is the classification between subject groups based on the fMRI signal variations. One method of accomplishing this is through functional network connectivity (FNC), an established set of methods applied within fMRI to infer differences in brain connectivity that may depend on a disease or mental state (van de Ven [2004]). More generally, temporal connectivity has been used to explore directed influences between neuronal populations in fMRI data (Roebroeck et al. [2005]) using Granger causality and to examine differences between schizophrenia patients compared to normal controls (Jafri et al. [2007], Garrity et al. [2007], Anderson et al. [2010]) using cross-correlation measures.

In this article we present code to perform functional connectivity analysis in **R** (R Development Core Team [2008]), that can be used either to discriminate between groups through a support vector machines (SVM) classifier or can be used to identify general connectivity differences between groups. The code presented here closely follows the methods presented in Anderson et al. [2010], which describes in full the motivation for and findings of using connectivity measures based on independent components analysis (ICA) Hyvärinen and Oja [2000] to classify between schizophrenia patients and normal controls during rest. We assume the reader is familiar with both fMRI data and ICA techniques. We also assume the reader has no specific knowledge of R, but does have general knowledge of basic programming techniques. We hard-code as little as possible; the 4 lines that will have to be changed for a new users' analysis are denoted with comments using the "#" symbol. The variables that will need to be adjusted are the list of NIFTI/ANALYZE fMRI scans contained in the file named "filenames.txt" and a list of labels for each scan. The user is encouraged to have all the files in a common directory. We assume the scans have already been preprocessed with motion correction using tools such as FLIRT within FSL. As this analysis focuses on connectivity within subjects, spatial alignment across subjects is not necessary. The **analyzefMRI**, **vegan**, **igraph**, and **e1071** are used along with their dependencies, and must be pre-installed. These packages are available at `http://cran.r-project.org/web/packages`.

The **R** platform has important benefits for fMRI analyses because of its availability and functionality. **R** is free and open source, so licensing costs for research are not prohibitive. It contains thousands of packages that can perform cutting-edge statistical techniques. **R** allows the user direct contact with their data, with routines for fMRI that can efficiently extract single timeseries, volumes, or planes. There are routines pre-built into **R** for fMRI that can perform methods such as mixed effects analysis, general linear models (GLM), Bayesian analysis and Granger Causality, in packages such as **analyzefMRI** (Bordier et al. [2009]), **MEMA** (Chen et al. [2010]), **cudaBayes** (da Silva [2010]), **FIAR** (Roelstraete and Rosseel [2011]), and **fmri** (Polzehl and Tabelow [2007]). We refer the reader to a presentation at the use-R! 2010 conference for a current description of some of the many packages and options in R for fMRI analyses, at `http://www.r-project.org/conferences/useR-2010/slides/Chen+Saad+Cox.pdf`. An **R** tutorial is available at `http://cran.r-project.org/doc/manuals/R-intro.html`.

This ability to access directly the data combined with the high-level statistical methods available within the general **R** framework offer the researcher much power in designing his own analysis, and easily allows the user to construct his own methods.

The code contained in this article is available through NITRC at willupdatewhenavailable. NITRC is an NIH-sponsored project to categorize, compare, rate and distribute software created by and for neuroimaging researchers. It contains both stand-alone programs and code snippets such as this project. Its usefulness is quite evident given the redundancies in research, where many labs develop independently routines to perform similar analysis techniques such as functional connectivity analysis. It is also useful for determining reproducibility, as users can test another's analysis on their own data to see if similar results are reached. This is particularly useful in fMRI analysis, where conclusions are often reached on quite small sample sizes since data are costly and difficult to obtain. The reader is encouraged to download and modify this code snippet from the NITRC website at `http://www.nitrc.org/projects/fmriclassify/`.

We perform classification by running independent components analysis (ICA) within **R** on the fMRI scans, and use the crosscorrelations of the components' timeseries to create a "distance" matrix. This distance matrix is then converted into a graph structure to compute the functional connectivity among independent components within each scan. We also demonstrate a distance matrix using timeseries from ICA output performed in FSL (Smith et al. [2004]). For both options, connectivity properties within each subject will give insight into networking differences attributable to either patient diagnosis or mental states. As the bulk of this code is constructed to classify between distance matrices, these routines can easily be adapted for an Region of Interest (ROI) analysis where distances are sought not between independent components, but instead between ROIs. Collectively, this article demonstrates code that can be adapted easily to new data for determining if functional connectivity differences exist between groups of fMRI scans, by altering only 4 lines of code.

2 Functional Network Creation

2.1 ICA for fMRI

Under the hypothesis that the activity of brain is constructed of networks acting together to produce meaningful psychobehavioral cognitive states, the aggregate activity is decomposed into subcomponents in ICA. An fMRI scan of time length T and spatial dimension S and can be expressed as a linear combination of $\tau < T$ components and the corresponding timeseries:

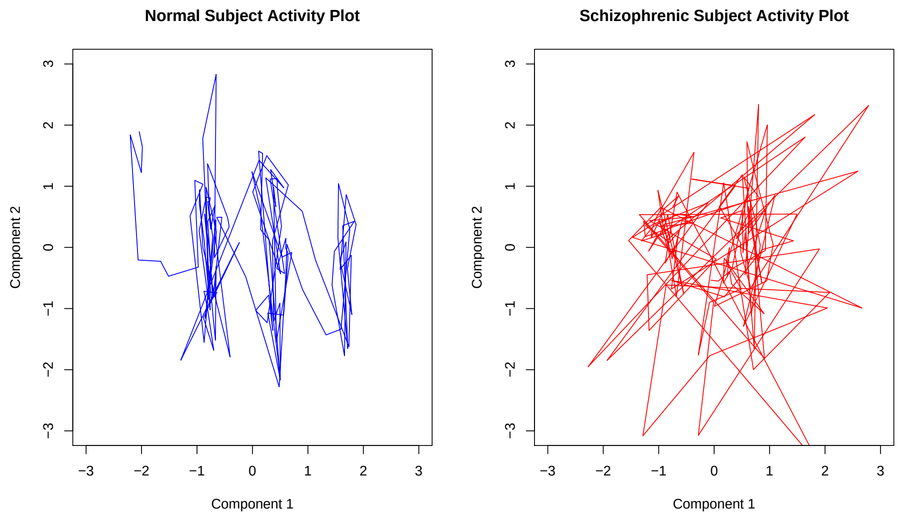
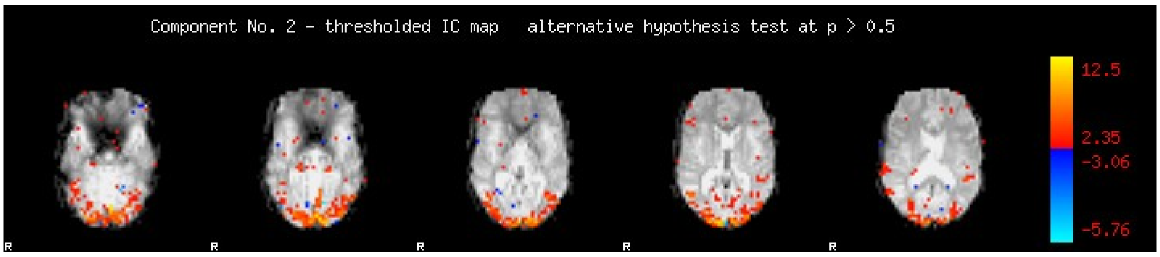


Figure 1: Temporal activity plot of two primary components within a subject, depicting the relationship between two components over time.



Component No. 2 - thresholded IC map alternative hypothesis test at p > 0.5
12.5
2.35
-3.06
-5.76

Figure 2: Spatial Map Produced by Independent Components Analysis in FSL

$$X_{ts} = \sum_{\mu=1}^{\mathcal{T}} C_{s\mu} M_{\mu t}$$

where X_{ts} represents the raw scan intensity at timepoint $t \leq T$ and spatial location $s \leq S$, $M_{\mu t}$ is the amplitude of component μ at time t, and $C_{s\mu}$ is the spatial magnitude for component μ at spatial location s. An example of a spatial map is shown in Figure 2.

To establish functional connectivity, the temporal associations between these independent components is computed and differences between groups are examined. A temporal interaction plot for a patient and normal controls is shown in Figure 1. The crosscorrelation function (CCF) between these timeseries is then calculated over a range of temporal lags, and the maximal cross-correlation is taken to be the similarity measure between two time-series. We subtract this cross-correlation value from 1 to create a pseudo distance measure, $d(M_\alpha, M_\beta)$, given by

$$d(M_\alpha, M_\beta) = 1 - max[CCF(M_\alpha, M_\beta, l)]$$

where

$$CCF(M_\alpha, M_\beta, l) = \frac{E[(m_{\alpha,t+l} - \overline{M_\alpha})(m_{\beta,t} - \overline{M_\beta})]}{\sqrt{E[(m_{\alpha,t} - \overline{M_\alpha})^2]E[(m_{\beta,t} - \overline{M_\beta})^2]}} \tag{1}$$

where $m_{\alpha,t+l}$ is time-shifted version $m_{\alpha,t}$, l is the time lag separating the two timeseries M_α and M_β, and $\overline{M_\alpha}$ is the mean of the entire timeseries $M_\alpha = (m_{\alpha,1}, m_{\alpha,2}, \ldots, m_{\alpha,T})$. The timeseries are calculated at lags ranging from 0 to 20% of the timeseries length, as higher lags results in fewer time points to calculate the correlation and a more noisy estimate. The distance function is then a transformation of the maximal absolute cross-correlation between two timeseries. The lag parameter is calculated over a maximum of 10 TRs using the option **lag.max**.

Two options are demonstrated for running ICA; the first option is to perform ICA within R, and requires a list of filenames in the current directory on which to run the ICA analysis. The second option is to create the functional networks using timeseries obtained from FSL, where the timeseries are contained in the **melodicmix** files found in the MELODIC routine (Smith et al. [2004]).

The current working directory can be obtained using **getwd()** This directory needs to contain the list of filenames to be operated on named **filenames.txt** and the set of scans. This directory can be navigated to using the command **setwd("/path/to/my/directory")**

2.2 Running ICA within R

We now read in our list of scans, and perform ICA on them. The filenames ("myfilename1.img") are written on a text file called filenames.txt, where each line contains a separate filename. The number of components within each subject (**numcomponents**) is an optional variable; we set the default value to 20 according to Smith et al. [2009].

```
rm(list = ls())
##################Change these lines##########################
setwd("/path/to/my/directory")
filenames<-read.table("filenames.txt")
##############################################################
####Call the library used for analysis.  This must be preinstalled.
library(AnalyzeFMRI)
numsubjects<-dim(filenames)[1]
###The number of components is set as default to 20.
numcomponents<-20
###Now create the array that will hold the "distance" matrix.
datadistance<-array(NA, c(numsubjects, numcomponents, numcomponents))
```

We then create a loop to read in our files, perform ICA on each scan using the function **f.ica.fmri** within the library **AnalyzeFMRI**, extract the timeseries associated with the components, and then create a distance matrix for each component.

```
###Run ICA on each scan and create a distance matrix
for (i in 1:numsubjects)
{   #Perform ICA on each file
     temp<-f.ica.fmri(as.character(filenames[i,1]),
     n.comp = numcomponents)
     for(j in 1:numcomponents)
     {   for(k in j:numcomponents)
          {     #Calculate the "distance".
                  datadistance[i,k,j]<-max(1-abs(max(ccf(temp\$A[,j],
                        temp\$A[,k], plot = FALSE, lag.max = 10)$acf,0)))
                datadistance[i,j,k]<-datadistance[i,k,j]
                }        }
           diag(datadistance[i,,])<-0}
s<-rep(numcomponents, numsubjects)
```

This code performs ICA on this i^{th} scan listed in the filenames of **filenames.txt**, and stores the results into an object called **temp**. This object has several attributes, one of which is the timeseries associated with each component called **temp$A**. The complete list of attributes can be identified using **attributes(A)**. The "distance" between the timeseries for each component is calculated and stored in **datadistance**. The diagonal is then set to be zero to correct for rounding errors, and the vector **s** is given the number of components within each subject to use for future analysis.

2.3 Using Melodic-FSL Output

Alternatively, if ICA has been run in an outside program such as FSL, the timeseries can also be read into **R** with the code below, which allows a variable number of components within each **melodicmix** file. The filenames **myfile1melodicmix** are stored in a text file called **melodicmixfiles.txt**:

```
##################Change these lines##########################
setwd("/path/to/my/directory")
filenames<-read.table("melodicmixfiles.txt")
##############################################################
###Find the numbers of components within each subject
s<-c(rep(0,numsubjects))
for (i in 1:numsubjects)
{        s[i]<-dim((read.table(as.character(filenames[i,1]))))[2]        }
```

This first block determines the number of components within each file, as FSL computes that parameter uniquely within each scan. This number is stored within the vector, **s**.

We next read in the timeseries and use these to create a distance matrix with the same distance metric as in the previous section.

```
###Now read in the data and create the distance matrix.
datadistance<-array(NA, c(numsubjects, max(s), max(s)))
for (i in 1:numsubjects)
{    ##Read in ICA results
     temp<-as.matrix(read.table(as.character(filenames[i,1])))
     for(j in 1:s[i])
    {       for(k in j:s[i])
        {     #Calculate the "distance" between two components.
                datadistance[i,k,j]<-1-abs(max(ccf(temp[,j],
                        temp[,k], plot = FALSE, lag.max = 10)$acf))
                datadistance[i,j,k]<-datadistance[i,k,j]
                        }       }
     diag(datadistance[i,,])<-0               }
```

2.4 Overview of Functional Network Creation

The previous section demonstrated how to create matrices capturing the functional network connectivity within a subject. At this point, all the fMRI scans have had ICA performed on them, and have each been reduced into a matrix where each element is the “distance” between two components. We next extract properties of these matrices to be used for classification. We computed the distance between components using maximal absolute cross-correlation as an indicator of the amount of information shared between them and how similarly they act over time. This is a flexible measure however, and can be changed if there is reason to believe that another metric would be more useful.

3 Feature Extraction

Because the matrices are all created within each subject, the rows and columns do not map across each subject. The matrices may also be different sizes. An example of this is shown in Figure 3.

To extract common features, we treat each matrix as representing a set of points on a high-dimensional manifold, where each point is an independent component and the distance between two points measures the similarity of their temporal activity. We then transform each matrix into a graph

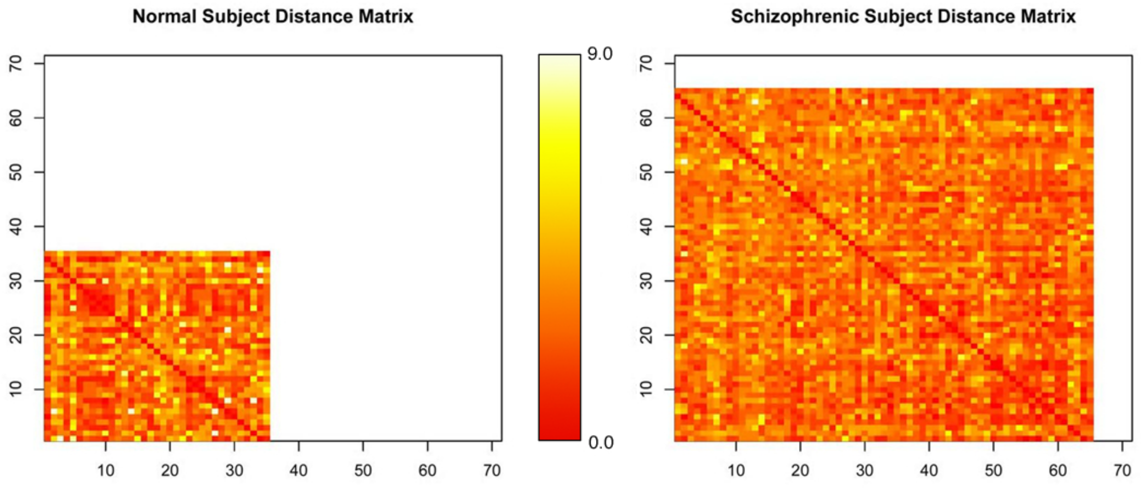


Figure 3: Distance Matrices, where rows and columns correspond to components within a subject and the intensity represents the functional connectivity between those components.

structure using the ISOMAP procedure, where a low-dimensional projection of the points is made using a geodesic distance calculation. This allows a linear decomposition such as multi-dimensional scaling to be performed on data that may not be linear, by computing distances between points as being the "path distance" across connected points. This would be equivalent to calculating the driving distance between two cities via connected roads, instead of measuring the distance as the crow flies, as shown in Figure 4.

Conceptually, each matrix is initially a set of connected points, where every point is linked to every other point even if they are dissimilar in their behavior with low functional connectivity. This imposes an inefficiency that may not be realistic. Creating "graphs" out of each matrix using a non-linear distance metric not only allows for a more efficient low-dimensional projection of the matrix, but also encourages the graph to be connected more efficiently by trimming poor connections while maintaining stronger ones. This fragmentation allows us to determine how strong connections are within the subject, how many subnetworks (subgraphs) exist, what the sizes of these subnetworks are, and how efficiently the points are connected overall. These properties, all interrelated, give quantitative measurements of the connectivity that can be used to fingerprint the networking associated with different disorders. An example these graphs is shown in Figure 5.

This section contains code to first transform each matrix into ISOMAP graph structure and measure the connectivity properties of each graphs. These connectivity measures eventually form a feature matrix for classification.

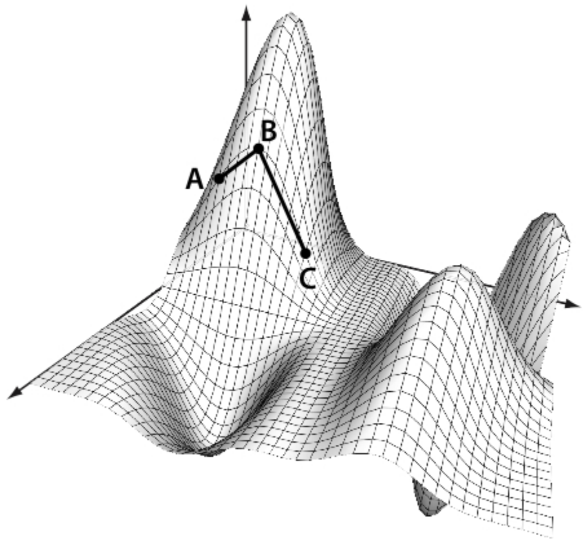


Figure 4: Geodesic Distance Calculation: The distance between A and C is calculated as the manifold path distance from A to B to C instead of directly from A to C.

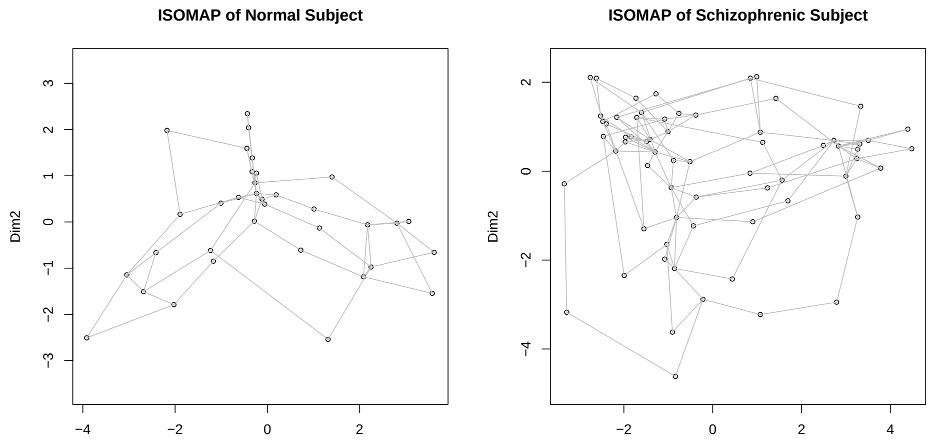


Figure 5: Graphs representing connectivity of two subjects

3.1 ISOMAP Embedding and Feature Extraction

The ISOMAP procedure uses the library **vegan** which must be preinstalled. We also use the package **igraph** which calculates different connectivity properties of the graphs.

```
library(vegan)
library(igraph)
```

We next create a function that takes an ISOMAP object and turns it into a graph structure. This function will be called when we create our distance matrix within a loop.

```
###Function to turn each ISOMAP object into a graph
makegraph<-function(myiso)
{     ##dim is dimension of matrix
     my_dist<-as.matrix(dist(myiso$points[]))
     k<-dim(my_dist)[1]
     mynet<-matrix(0, nrow = k, ncol = k)
     which.rows<-myiso$net[,1]
     which.cols<-myiso$net[,2]
     for(j in 1:length(which.rows))
        {   mynet[which.rows[j], which.cols[j]] <-my_dist[which.rows[j], which.cols
        mynet[which.cols[j], which.rows[j]] <-my_dist[which.cols[j], which.rows[j]]
        mynet
}
```

We next run our ISOMAP procedure within each distance matrix, covert the output into a "graph" format, and create a feature matrix that contains measures of the directed graph's properties.

```
myfeaturematrix<-matrix(NA, nrow = numsubjects, ncol = 10)
for(i in 1:numsubjects)
{       d<-matrix(datadistance[i,1:g[i],1:g[i]], nrow = g[i])
        myiso<-isomap(d[1:g[i],1:g[i]],axes=1, epsilon=median(d), ndim = 10)
        mynet<-makegraph(myiso)
        d2 <- graph.adjacency(mynet, weighted = TRUE )
        myfeaturematrix[i,]<-c(average.path.length(d2),clique.number(d2),
           graph.density(d2),edge.connectivity(d2),median(closeness(d2)) ,
           median(degree(d2)),max(degree(d2)),vcount(d2),ecount(d2),
                   transitivity(d2))
        }
```

These measures are:

- Average Path Length **average.path.length()**: Average path length between all connected vertices.
- Clique Number **clique.number()**: Number of elements in the largest subgraph.
- Graph Density **graph.density()**: Ratio of the number of edges to the number of possible edges.
- Edge Connectivity **edge.connectivity()** (also called graph adhesion): Minimum number of edges needed to obtain a graph which is not strongly connected.
- Median Closeness **median(closeness())**: Median number of steps required to access every other vertex from a given vertex.
- Median Degree **median(degree())**: Median number of edges incident to a vertex, with loops being counted twice.
- Maximum Degree **max(degree())**: Maximum number of edges incident to a vertex, with loops being counted twice.
- Vertex Count **vcount()**: Number of Vertices in the graph
- Edge Count **ecount()**: Number of Edges in the graph
- Transitivity **transitivity()**: Probability that two vertices are connected.

Finally, we "label" the columns.

```
colnames(myfeaturematrix)<-c("Average Path Length", "Clique Number",
    "Graph Density", "Edge Connectivity", "Median Closeness",
    "Median Degree", "Max Degree", "Vertex Count", "Edge Count",
    "Transitivity")
```

3.2 Overview of Feature Extraction

In this section we have converted each distance matrix into a feature vector. The matrices were individually created into a graphical structure of connected points using a non-linear distance metric. Properties of this graphical structure that measure the connectivity are extracted into the feature vector. All of these measures are correlated; for example, a graph with a low "median closeness" measure would imply that there is a short distance between two vertices, thus increasing the transitivity. Because of this, multiple comparisons must be adjusted for within any analysis that relied on formal hypothesis testing. The feature vectors collectively form a feature matrix that will be used for classification in the following section.

4 SVM Classification

In this section we demonstrate SVM Classification using the package **e1071** that contains an interface to the libsvm **C++** package by by Chih-Chung Chang and Chih-Jen Lin. The **R** vignette (`http://cran.r-project.org/web/packages/e1071/vignettes/svmdoc.pdf`) details the functionality of this package, which includes many other classification routines besides SVM. The SVM method within this package has an optional benefit of cross-validation, which simplifies coding dramatically by implementing the training and testing steps within a single function call. In the following code we demonstrate classification of our feature vector using 10-fold cross-validation, but this is a parameter that can be changed.

We create a vector **mycategory** with the response variables, in the case the patient diagnosis of each scan. This can alternatively be read in using the function **read.table()**. There are many options within the **svm()** method that can be specified such as kernel choices, but we use the default parameters here ("radial basis function") for the sake of conciseness and clarity.

```
library(e1071)
##################Change these lines#########################
mycategory<-c(rep("Normal",17), rep("Schizophrenic",10))
mycategory<-as.factor(mycategory)
#############################################################
mysvm<-svm(mycategory~., data = myfeaturematrix, cross=10)
mysvm$accuracies
```

The structure **mysvm** contains many details of the model. We can see the average cross-validation accuracy within each k^{th}-fold using **mysvm$accuracies**.

5 Conclusion

Collectively, we have demonstrated code to determine if functional connectivity differences exist between groups. Functional connectivity analysis has been demonstrated to give discriminatory power in classifying Schizophrenia and Alzheimer's disease patients, and performing this analysis in **R** allows the user flexibility in altering such as distance metrics, classification machines, and feature selection choices. This code demonstrates SVM classification in **R**, a "black-box" model which ironically is remarkably simple with a single function call to both train and test the model using cross-validation. Because **R** is an established package in the statistics research community, many newer procedures can easily be implemented to compare with more established classification machines.

Although this analysis was created for analysis of fMRI data, more generally it applies to problems where the relationship among signal sources may determine the category to which an object belongs. The joint behavior of the signal sources (independent components) was observed as a graph object, where the distances between the sources represented the similarity of their behavior. Although second order measures were used to assess the functional connectivity (correlations), it is possible that as much discriminatory power exists using higher-order measurements that take into account the cohesiveness of triplets of components, or even more. Functional connectivity is metric that should be assessed from multiple angles, and distributing code to do this might encourage the neuroimaging community to do so.

6 Acknowledgements

We thank Dr. Xavier Quintana for providing data and guidance during the original analysis and NIH grant number R33DA026109 to MSC.

References

A Anderson, ID Dinov, J Sherin, JE Quintana, AL Yuille, and MS Cohen. Classification of spatially unaligned fMRI scans. *NeuroImage*, 49:2501–2519, 2010.

C. Bordier, M. Dojat, and P. Lafaye, de Micheaux. Analyzefmri: an r package to perform statistical analysis on fmri datasets. *UseR 2009*, 2009. URL `http://www.biostatisticien.eu/AnalyzeFMRI/`. Software: R Package, AnalyzeFMRI, version 1.1-12.

Gang Chen, Ziad S Saad, and Robert W Cox. Statistical analysis programs in r for fmri data. *in press*, 2010.

AF da Silva. cudabayesreg: Bayesian computation in cuda. *The R Journal*, 2:48–55, 2010.

A. G. Garrity, G. D. Pearlson, K. McKiernan, D. Lloyd, K. A. Kiehl, and V. D. Calhoun. Aberrant "default mode" functional connectivity in schizophrenia. *Am J Psychiatry*, 164(3):450–457, March 2007. ISSN 0002-953X. doi: http://dx.doi.org/10.1176/appi.ajp.164.3.450. URL `http://dx.doi.org/10.1176/appi.ajp.164.3.450`.

Aapo Hyvärinen and Erkki Oja. Independent component analysis: Algorithms and applications. *Neural Networks*, 13(4–5):411–430, 2000. URL `http://www.cis.hut.fi/~{}aapo/papers/IJCNN99_tutorialweb/`.

Madiha J. Jafri, Godfrey D. Pearlson, Michael Stevens, and Vince D. Calhoun. A method for functional network connectivity among spatially independent resting-state components in schizophrenia. *Neuroimage*, November 2007. ISSN 1053-8119. doi: 10.1016/j.neuroimage.2007.11.001. URL `http://dx.doi.org/10.1016/j.neuroimage.2007.11.001`.

Jörg Polzehl and Karsten Tabelow. fmri: A package for analyzing fmri data. *RNews*, 7(2):13–17, 2007. URL `http://www.r-project.org/doc/Rnews/Rnews_2007-2.pdf`.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

Alard Roebroeck, Elia Formisano, and Rainer Goebel. Mapping directed influence over the brain using granger causality and fmri. *NeuroImage*, 25(1):230–242, March 2005. doi: 10.1016/j.neuroimage.2004.11.017. URL `http://dx.doi.org/10.1016/j.neuroimage.2004.11.017`.

Bjorn Roelstraete and Yves Rosseel. Fiar: An r package for analyzing functional integration in the brain. *in press*, 2011.

Stephen M. Smith, Mark Jenkinson, Mark W. Woolrich, Christian F. Beckmann, Timothy E. J. Behrens, Heidi Johansen-berg, Peter R. Bannister, Marilena De Luca, Ivana Drobnjak, David E. Flitney, Rami K. Niazy, James Saunders, John Vickers, Yongyue Zhang, Nicola De Stefano, J. Michael Brady, and Paul M. Matthews. Advances in functional and structural MR image analysis and implementation as FSL. *NeuroImage*, 23:208–219, 2004.

Stephen M. Smith, Peter T. Fox, Karla L. Miller, David C. Glahn, P. Mickle Fox, Clare E. Mackay, Nicola Filippini, Kate E. Watkins, Roberto Toro, Angela R. Laird, and Christian F. Beckmann. Correspondence of the brain's functional architecture during activation and rest. *Proceedings of the National Academy of Sciences of the United States of America*, 106 (31):13040–13045, August 2009. ISSN 1091-6490. doi: 10.1073/pnas.0905267106. URL `http://dx.doi.org/10.1073/pnas.0905267106`.

V. G. van de Ven. Functional Connectivity as Revealed by Spatial Independent Component Analysis of fMRI Measurements During Rest. *Human Brain Mapping*, 22:165–178, 2004.