ECE 532 : Using fMRI derived functional network features for disease and stimuli classification

Sam Rusk, Aritra Biswas, Chris Fernandez

December 13, 2014

1 Introduction

Functional magnetic resonance imaging (fMRI) is an imaging technology which is used clinically and in research to investigate the spatiotemporal dynamics of neural activity in the brain. The measured signals represent a Blood Oxygen Level Dependent (BOLD) response that corresponds to functional neural activations. In resting state fMRI, subjects relax and steady state functional activations can be observed in the brain. Analysis of these functional activations allows us to model the interactions of individual brain networks, whose activity patterns can vary significantly as a result of a neurological or psychiatric disease. Another type of experiment is the so called block design. In these experiments, patients are presented with a series of stimuli, and their neurological response to these stimuli are measured. Each individual stimulus and corresponding neural activation signal comprise a block.

Our goal is to apply Machine Learning tools learned in this class to implement a classifier that can make accurate predictions about a patient based on functional network features derived from fMRI images. In this lab, we will go through two prediction exercises. In the first, we will attempt to diagnose if a patient has Schizophrenia or not, using functional network features derived from fMRI images. In the second exercise, we will classify the type of stimuli a patient received as visual or auditory based on their fMRI data; for example, we will try to predict whether a patient has seen an image of a frog or heard its 'ribbit' instead.

2 Overview

After image data is acquired and transformed into image space, fMRI data consists of a sequence of 3-dimensional images, with a corresponding BOLD signal for each voxel over time; this is essentially a 3D video of brain activations over time. A voxel is the three dimensional unit of volume in fMRI data and it represents one element in the 3-D coordinate system.

Before we can classify our data we need to perform feature extraction and feature selection. The fMRI data in its raw processed form has hundreds of

thousands of voxels over hundreds of time intervals. We do not have enough time or computer memory to perform analysis over so many voxels.

Furthermore, when performing a specific activity, not all parts of the brain are active. The process of feature extraction involves selecting the active voxels and thus reducing the dimensions of our data matrix. Sometimes even feature extraction doesn't reduce our data substantially to perform classification operations on it. We then use a technique called feature selection where we select a subset of the active voxels to make a prediction. Finally after we have reduced the raw data matrix to a usable rank; we can implement a training algorithm to build a classifier. For this lab, the feature extraction and selection has already been performed for you. The following section describes these processes in more detail for each dataset.

3 Datasets

The first dataset was used for the official competition at the 2014 IEEE International Workshop on Machine Learning for Signal Processing, aptly called the MLSP 2014 Schizophrenia Classification Challenge. The data matrix is 86 by 410 in size, containing 410 features (columns) for 86 total patients (rows). These features represent the current state-of-the-art developments in neuroimaging data processing, and incorporate both functional and structural information. The first 379 columns contain the functional network connectivity (FNC) data, obtained using resting state fMRI on a cohort of schizophrenic and healthy patients. For the feature extraction step, group independent component analysis (GICA) was performed on the entire cohort. GICA decomposition of fMRI data results in a brain map of spatial components that are consistently active in the entire cohort. Components are essentially blobs consisting of many voxels that are spatially and functionally related within themselves, but try to maximize statistical independence from all other brain components. An example component representing the default mode network is presented in figure 1.

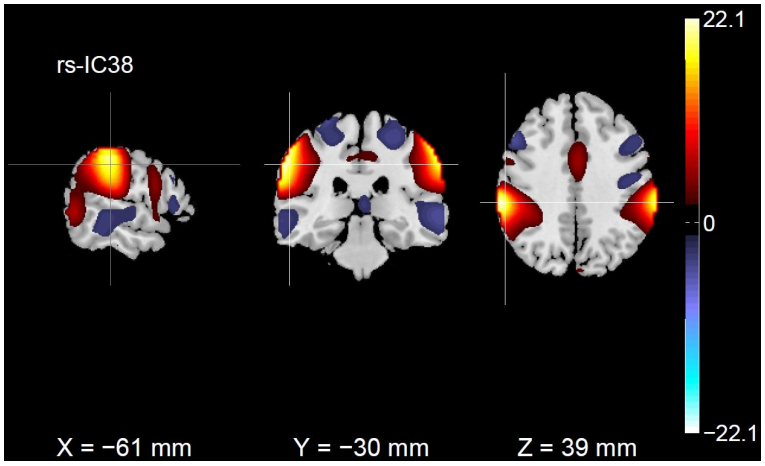

Figure 1: A component map representing part of the Default Mode Network

After selecting a subset of 28 components that most closely resemble well

studied brain networks, the correlation of time series from each component, to every other component is computed, resulting in a 28 by 28 correlation matrix for each individual subject. Half of the matrix is redundant (as the matrix is symmetric about its diagonal), so all of the unique components are then moved into a vector form, providing us with the first 379 feature columns for each patient in our data matrix. 31 Structural fMRI features are then appended to each of these vectors, providing a the final 410 features our data matrix.

The second dataset was collected for an fMRI experiment performed here at UW. This experiment followed a block design where participants were presented with either an auditory cue (e.g., a "ribbit" to reference a frog, or a "woof" to reference a dog) or a visual cue (a black and white line drawing, depicting just the concept without supporting context). In total, 37 concepts were presented as stimuli. The participant saw the visual cue 4 times and heard the auditory cue 4 times for each concept, giving 37 * 4 * 2 = 296 trials, which corresponds to the number of rows in the data matrix. The data are organized in the following way; the first 148 rows are the visual presentations, and the second 148 rows are the auditory presentations. The concepts are arranged in alphabetical order, and loop every 37 rows. So, row 1 is airplane, row 38 is airplane, row 75 is airplane, etc. For each subject's dataset, there are voxel coordinates, a visual or auditory label, and a specially processed BOLD signal value, as shown in figure 2. Here, the blue line is the "raw" fMRI signal intensity in a single voxel, the red line step function indicates when a stimulus is presented to a subject, and the green line is the red line convolved with some function that approximates the canonical neural response function. The height of that green curve is controlled by a single parameter, and this is the value that we report for the voxel.

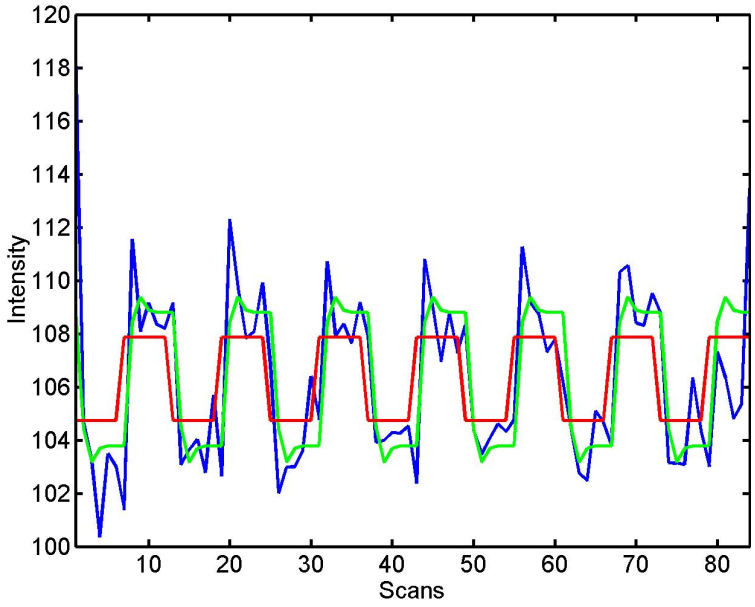

Figure 2: Graph showing the feature extraction steps for one voxel

4 Warmup

All solutions should be implemented in the ECE532 fMRI Classification lab matlab file. Some code has been provided to get you started.

4.1 The least squares algorithm for underdetermined systems.

Almost all fMRI or gene expression data we deal with are underdetermined. Underdetermined solutions could have infinitely many solutions to a least square problem. So a weights vector that fits well to a training set might perform terribly against a different test set. Thus in order to get a good estimate of a weights vector that is uniformly spread over all the columns of the data matrix; we solve the least squares problem and take the weight vector with the smallest L2 norm

Try out the following code for solving the least squares problem using the Moore pseudoinverse method. Observe the resulting bhat. How does it compare to the true b vector?

```
%% Moore Pseudo inverse method
clear all; close all; clc;
% Random data matrix A with 1000 points with 2 features
% Labels vector b for A
m = 1000;
n = 2;
b = zeros(m,1);
for i=1:m
    a = 2*rand(2,1)-1;
    A(i,:)=a';
    b(i) = sign(a(1)^2+a(2)^2-.5);
end

% Getting the pseudo inverse
pseudo_inverse = pinv(A);
% Calculating the weights vector
x_hat = pseudo_inverse*(b);
% Compute the prediction vector
b_hat = sign(A*x_hat);

% Visualize the unclassified data
figure(1)
hold on
plot(A(:,1),A(:,2),'.');
axis('square')
title('Data Matrix A')
hold off
% Visualize the classified data
figure(2)
hold on
for i=1:m
    a = A(i,:);
    if b_hat(i)==1
        plot(a(1),a(2),'b.');
```

```
    else
        plot(a(1),a(2),'r.');
    end
end
axis('square')
title('More Pseudoinverse LS Classifier')
hold off
```

4.2 The Regularized least squares

In the Regularized least squares algorithm we account for some noise present in the data. Real world data is not perfect and we often face the problem of overfitting the classifier to the data. The higher the value of lamda the more we accomodate for noise. A lamda value of zero if the same as doing regular least squares. The following section of code goes through how to solve for the weights vector using regularized least squares.

Test out this code on the same data as 4.1. Are the results different? How might different regularization parameters change our classifier?

```
%% Regularized Least Squares Method

% Random data matrix A with 1000 points with 2 features
% Labels vector b for A
m = 1000;
n = 2;
b = zeros(m,1);
for i=1:m
    a = 2*rand(2,1)-1;
    A(i,:)=a';
    b(i) = sign(a(1)^2+a(2)^2-.5);
end

% Calculating the SVD of A; the econ parameter aligns
% the dimensions of U,D,V properly
[U,D,V] = svd(A,'econ');
% the error term lambda
lambda =0.1;
n = length(D);
x_hat = V*inv(D'*D + lambda*eye(n,n))*D'*U'*b;
b_hat = sign(A*x_hat);
% NOTE: sometime the matlab inv function can be slower and inaccurate
%       for large data sets. If a matrix is a square diagonal matrix;
%       it's inverse can be calculated by inverting the diagonal elements

%Alternate method to compute inverse
Dinv = zeros(n,n);
for i=1:n
    sum = D'*D + lambda*eye(n,n);
    Dinv(i,i)= 1/sum(i,i);
end
temp = inv(D'*D + lambda*eye(n,n));
```

```
%check if the two are equal or not:
if (Dinv) == (temp)
    disp('These methods are equal!');

% Visualize the classification results
figure(3)
hold on
for i=1:m
    a = A(i,:);
    if b_hat(i)==1
        plot(a(1),a(2),'b.');
    else
        plot(a(1),a(2),'r.');
    end
end
axis('square')
title('Regularized LS Classifier')
hold off
end
```

4.3 Support Vector Machine

In machine learning, support vector machines (SVM's) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier.

Try out the following SVM implementation in Matlab and observe the results.

```
%% SVM
% generate test data
mtest = 1000;
ntest = 2;
btest = zeros(mtest,1);
for i=1:mtest
    a = 2*rand(2,1)-1;
    Atest(i,:)=a';
    btest(i) = sign(a(1)^2+a(2)^2-.5);
end
%generate train data
m = 1000;
btrain = zeros(m,1);
for i=1:m
    a = 2*rand(2,1)-1;
    Atrain(i,:)=a';
    btrain(i) = sign(a(1)^2+a(2)^2-.5);
end

%train
```

```
SVMStruct = svmtrain(Atrain,btrain,'kernel_function','polynomial');
%predict
bhat = svmclassify(SVMStruct,Atest);

%visualize
figure
subplot(121); hold on;
for i=1:mtest
    a = Atest(i,:);
    if btest(i)==1
        plot(a(1),a(2),'b.');
    else
        plot(a(1),a(2),'r.');
    end
end
axis('square')
title('Test data')
subplot(122); hold on;
for i=1:mtest
    a = Atest(i,:);
    if sign(bhat(i))==1
        plot(a(1),a(2),'b.');
    else
        plot(a(1),a(2),'r.');
    end
end
axis('square')
title('SVM')
```

If you look closely, you'll realize that the training and test data are generated in the same randomized way as in our previous two examples; this means that we didn't see the full picture before. This is because the SVM uses a nonlinear kernel to separate the data in a higher dimensional space. This aspect of the kernel SVM proves useful for many prediction problems.

5 Schizophrenia prediction with rs-fMRI data

Now that we have warmed up with our basic classifiers, we can now apply these tools to our first clinical problem. Note that the fMRI data matrix A may be still be underdetermined, but nonetheless, some of the tools mentioned in the warmup should enable you to produce meaningful predictions. For the sake of notation A is mxn and b is mx1.

5.1 Loading in the data

The data matrix and labels matrix for both datasets are be provided in the lab subfolder called kaggledata. Normally these values must be extracted from the images and restructured for analysis. To load up this data into A, Atest, and b, run the code in section 5.1 of this exercise.

5.2 Implementing the Classifiers

Now that we have loaded the data, implement the Moore Pseudoinverse Least Squares, Regularized Least Squares, and SVM classifiers outlined above. Use the full training set to train each model.

5.3 Testing and Error Estimation

The error is measured by counting the number of wrong predictions made by the classifier. We compare our predictions to the labels vector of the test data to check for accuracy. Kaggle is the name of a popular website that crowdsources data science and machine learning competitions. Since this is competition, Kaggle has inflated the number of rows in the test set to create a much larger data sample so that cheating is harder. As such, we must upload our outputs to Kaggle for error estimation and scoring. Fortunately, this is very easy and fast, and can be done in about 30 seconds by running the code in section 5.3 in our lab file, and uploading the output .csv file to the provided link.

Using this submission model, you should be able to benchmark each of your classifiers, and iterativly test changes to them as you seek to maximize your predictive accuracy. Record the highest benchmark you achieve, and describe the classifier and any additional steps you utilized in your method. Hypothesize about the properties of this dataset that may enable your best implemented method to be the most robust.

6 Block Design Stimuli Recognition

Now that you have some basic experience with fMRI classification, we'll dive a bit deeper with our Block Design data set as we try to differentiate visual from auditory stimulus using only brain signals. Load in "blockdesigndata.mat" in the same way as in section 5. You should now have a data matrix "X" that is 296x26155 in dimensions - a highly underdetermined system. In addition, you should also find your "isVisual" label vector, denoting whether or not a stimulus is visual with a 1, or auditory with a 0.

You will notice that this time, the training and test data have not been split up for you. This is so that we can gain some experience generating training and test subsets from a single dataset, a very common practice in machine learning research. In the next sections, we will split up the data into k testing examples and m-k training examples. The two most common approaches to splitting the data up are described below. Each row of the data set is a training or test example.

6.1 Partitioned-Subset Cross Validation

The fist method involves splitting the data into fixed sizes or groups. For example if m=100, we could split the data up into 10 groups of size 10. The size of a group is the number of examples in the group. Once we have grouped the data up, we hold out one group as the test group and use the remaining as training data. We try all possible combination of testing and training groups and average the error over all the trials. In the above example we would have

10 possible test and training sets. If we selected two groups to test instead we would have 10 choose 2 possibilities to sample over.

Implement your own partitioned-subset cross validation function to generate test and training sets from our Block Design data. More specifically, try splitting the data into 8 groups, 16 groups, and 32 groups, holding out only 2 groups for testing in each scenario. Some of these group sizes will are not exactly divisible; in these cases, take the floor of your grouping and add the remainder subjects into the test dataset. Test out the 3 classifiers described above using this cross validation approach. What happens when we use a proportionately larger training set? Which CV approach gives us the best results? Consider the downside of using a smaller test set.

6.2 Randomized Cross Validation

The second method involves randomly selecting k test examples and setting the remaining m-k rows as training examples. Once we have a the data split up. We build the weights vector using one of the above methods on the training data and then check for accuracy on the testing data. We perform this randomization p times and average the net error over p samples to get an estimate of how our classifier is doing.

Repeat the exercise above using randomized cross validation instead of the partitioned-subset approach. How do your results compare? Which individual trial and classifier gave the overall best results? Experiment with a couple values for p and comment on the effect of averaging net error over a larger number of cross validation folds.

6.3 Interpretation of results

For your best performing classifier, take a look at the bhat elements that were predicted incorrectly, keeping in mind that the patient saw a visual cue 4 times and heard the auditory cue 4 times for each concept, giving 37 * 4 * 2 = 296 trials, corresponding to each element of bhat. If any specific concepts appear to be more wrongly predicted more than others, comment on any qualitative relationships those concepts have with one another. The goal here is just to see if, and how, any of our chosen stimuli concepts may affect classification performance; there are no right or wrong answers here.

7 A Neuroscience Perspective: using feature selection methods to study functional networks

If implemented appropriately, you should have been able to achieve at least a 7 in 10 prediction accuracy in both of the previous experiments. While this is certainly not accurate enough to use clinically as a diagnostic tool, there is still information in the data to form meaningful conclusions. One example of this is the use of the LASSO (least absolute shrinkage and selection) algorithm. We can use the LASSO algorithm to predict which features in the data matrix are most important for the classification problem.

The LASSO algorithm zeros out the unimportant columns in the data matrix and keeps only the ones it deems to be important. Thus LASSO can also be used

as a column subset selection algorithm or dimensionality reduction technique. The following code snippet generates a sparse lasso vector. The indices of this vector refer to the columns of the data matrix.

Run the following LASSO analysis on the kaggle Schizophrenia dataset. Note that you may need to rerun code section 5.1 to regenerate the schizophrenia data matrices and vectors.

```
% numsamp is the number of columns of matrix A
% A is the data matrix
% b is the labels vector

x = zeros(length(A),1);
alpha = 1/trace(A'*A);
lambda = 25;
delta = 100;
while delta > .001
xtmp = x + alpha*A'*(b-A*x);
xnew = sign(xtmp).*max(abs(xtmp)-alpha*lambda,0);
delta = norm(x-xnew);
x = xnew;
end

% x is the lasso vector
key_networks = find(x~=0);

% x is the lasso vector
```

Observe the size of the key networks vector. This tells us that of all 410 features, only 37 are the most discriminative between schizophrenia and healthy subjects. Neuroscientists often use this or similar methods to further reduce the dimensionality of their data and to hone in on the most critical features. In this dataset, each of these 37 features represents a dynamic connectivity between two well understood brain networks; for example, the visuomotor to attentional network connection. By identifying these between-network connectivities, neuroscientists are able to probe individual networks at a deeper level to better understand and elucidate factors for psychiatric disease.

8 Conclusion

In conclusion, machine learning methods can prove to be very robust tools for understanding complex neural functions, and have many practical applications from diagnosis to reverse engineering brain networks. Exciting new applications of machine learning are present in the areas of brain-computer interfaces, mapping of the functional connectome, and many others. Moreover, concepts from neuroscience have a rich history of providing new ideas and frameworks through which we can create new machine learning algorithms and architectures. Recent results in the application of representation and deep learning to classify large scale image datasets provides a strong example of this cross-pollination in practice.

9 References and Further Reading

1. https://www.kaggle.com/c/mlsp-2014-mri/data
2. http://cercor.oxfordjournals.org/content/early/2012/11/09/cercor.bhs352.abstract
3. http://www.ncbi.nlm.nih.gov/pubmed/22470337
4. http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=1385344
5. http://papers.nips.cc/paper/3660-discriminative-network-models-of-schizophrenia.pdf
6. http://neuro.engr.wisc.edu/program.html
7. http://www.math.duke.edu/ ingrid/publications/IComponent.pdf