Classifying brain states induced by complex visual stimuli

Andrew Floren
1 University Station, C0803
The University of Texas at Austin
Austin TX, 78712-1084 USA
afloren@mail.utexas.edu
(214) 384-2895

Abstract

abstract

1 Introduction

In traditional fMRI experiments, investigators seek to identify relationships between the measured BOLD signal and a carefully designed stimulus in order to tease out the purpose of particular brain regions. Recently, a new trend has developed where researchers are instead looking to predict what stimulus was presented given the measured BOLD signal [2, 5, 3]. While successful, most of these experiments involve presenting static images from a limited number classes such as faces and places. Then, the researchers try to classify which image or class of images was presented during each frame by analyzing the measured BOLD signal using machine learning classifiers. While this has proven to be a successful approach, it does not mimic the dynamic environment in which brains have evolved. Our goal is to analyze brain function with dynamically changing stimuli that portray real world experiences. Further, we were interested to see what information could be gleaned from the BOLD signal beyond object categories. We used virtual world technology to specify the stimuli in detail. Given our long-term interest in PTSD, we created a virtual town intended to suggest the kinds of real-world settings currently encountered by our military forces. Virtual character representing friendly forces and hostile combatants were presented in a virtual town. We then trained linear SVMs (support vector machines) and feed forward neural networks to predict the number of characters in each stimulation. [Bruce: this is most of your introduction from the abstract but feel free to rewrite the whole thing. I'll add in the literature review once you are happy with the introduction.]

2 Methods

2.1 Stimulus

We developed a virtual reality environment similar to many popular first person video games using the Unreal Engine 2 SDK [**?**]. The stimuli is dynamically rendered and presented from the point of view of a camera moving through this virtual environment while characters are presented. The stimulus employs a classic block design, in which the viewpoint moves for 15 seconds through the virtual environment (an example frame is presented in figure 1), then pauses for 15 seconds during which a group of characters fades into view (an example frame is presented in figure 2). Each run consists of 12 alternations between moving through the virtual environment and character presentations. In each scanning session, 4 to 6 runs were collected.

The camera is constantly moving (even during the character presentation periods the camera slowly pans and rotates) and the characters are animated so that the presented scene is never static. The number and location of characters varies with each presentation in a quasi-random fashion. The number of characters presented varied from 1 to 6. A presentation with a particular number of characters appears twice in each run, however the order of these presentations was randomized. It should be noted that this random ordering was generated once and held constant between subjects. Additionally, even between character presentations with the same number of characters, the locations of those characters varies considerably as seen in figure 3. [I need another frame from the stimulus to illustrate this point. These figures are all from the original report but I'd like to generate new ones if I can get access to a machine with the stimulus.]

Figure 1: An example frame from the stimulus while the camera is moving through the virtual envronment.

2.2 fMRI

We collected whole brain scans using a GRAPPA-accelerated EPI, with a 2.5 second TR and 2.5 mm cubic voxels on 40 slices that covered the majority of each subjects brain (see figure 4).

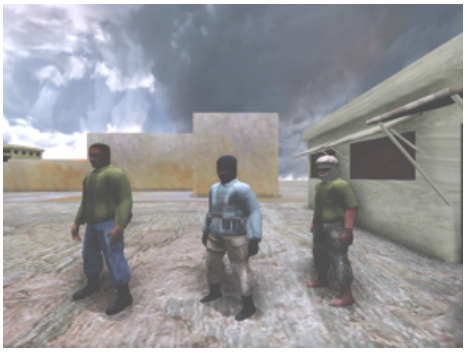

Figure 2: An example frame from the stimulus while characters are being presented.

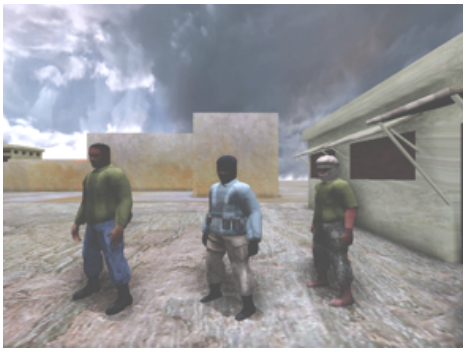

Figure 3: Two example frames depicting character presentations with two characters. The locations of the two characters varies considerably between the two frames.

Figure 4: An example prescription from one of the subjects.

2.3 Preprocessing

We performed motion compensation and slice timing corrections [**?**, **?**]. [David: you mentioned there were some good references for these two corrections.] Additionally, we applied a Wiener filter deconvolution [**?**] using a generic difference-of-gamma HRF [**?**] to shift the peak response in time so that it is aligned with the stimulus that caused it.

Wiener filter deconvolution can be summarized as follows: Given a system

$$y(t) = h(t) * x(t) + n(t) \tag{1}$$

where $x(t)$ is the signal of interest, $h(t)$ is some blurring kernel, $n(t)$ is independent additive noise, and $y(t)$ is the recorded signal. We want to find the deconvolution kernel $g(t)$ such that

$$\hat{x}(t) = g(t) * y(t) \tag{2}$$

minimizes the mean squared error between $x(t)$ and $\hat{x}(t)$, or

$$\sum_t \left(\hat{x}(t) - x(t)\right)^2 \tag{3}$$

The solution for the optimal $g(t)$ is most easily expressed in the Fourier domain.

$$g(t) \xrightarrow{\mathcal{F}} \frac{H^*(f)}{|H(f)^2| + \mathrm{SNR}^{-1}(f)} \tag{4}$$

Where $\mathrm{SNR}(f)$ is the signal to noise ratio $\frac{|X(f)|}{|N(f)|}$ at frequency f.

In fMRI, $y(t)$ is the recorded BOLD signal, $h(t)$ is the hemodynamic response function, and $x(t)$ is the neuronal population response. Calculating $g(t)$ requires estimates of the power spectral density of the signal of interest as well as the noise. Unfortunately, the noise function $n(t)$ corresponds not only to scanner noise but other nuisance factors as well. This makes modeling the noise, and thus its power spectral density, very difficult. Therefore, we have simply set $\mathrm{SNR}(f) = 1$ for all frequencies f. The primary effect of the resulting deconvolution is to shift the time series according to the delay caused by the hemodynamic response. Figure **??** illustrates the effects of this deconvolution on a simple square wave as well as an example voxel time series.

Finally, we reduce the dimensionality of the problem by masking out a subset of the volume using a harmonic power analysis. We selected the N (1000-3000) voxels with the greatest power at the frequency of the block alternation and its harmonics. In other words, we selected the voxels that responded in any fashion that covaried with the stimulus alternations. Let $y(t)$ be the recorded discrete time series at some voxel. Then let $Y(f)$ be the discrete Fourier transform of $y(t)$. The harmonic power of that time series is defined as:

$$\frac{\sum_{i=1}^{M} |Y(i \cdot N)|^2}{\sum_f |Y(f)|^2} \tag{5}$$

Figure 5: (1) The difference-of-gamma HRF employed as $h(t)$. (2) The deconvolution kernel $g(t)$ calculated from $h(t)$. (3) A simple square wave. (4) The same square wave after convolution with $g(t)$. (5) An example voxel time series. (6) The same time series after convolution with $g(t)$.

Where M is the number of harmonics and N is the frequency of interest, in our case the period of the block alternations. [David: feel free to elaborate on why this is a reasonable thing to do.]

Reducing dimensionality using the labels that are ultimately intended for classification is a good example of cross-contamination and will result in optimistic classifier performance estimates [**?**]. It should be noted that the harmonic power selection was based on the alternation between presentation of characters and the moving through the virtual environment, but without regard to the number of characters presented. Therefore, we will only be presenting classifier accuracy estimates for character count and not for the presence or absence of characters.

2.4 Classification

Using the time series from the voxels selected by the harmonic power analysis, we trained a linear SVM and a feed forward neural network. These algorithms were trained and validated using a cross-fold approach [4]. Each frame or point in the time series was treated as a separate data point instead of averaging across the block.

[Should I introduce the concept of training, validation and test sets here?]

Previous studies have discussed issues with optimistic performance estimates due to temporal correlations violating independence assumptions between training and test set examples [**?**]. We were interested in more closely examining the relationship between performance estimates and this temporal correlation. To accomplish this, we estimated classifier performance using 5 different methods

for splitting the training and test examples. These methods were designed to vary the average temporal distance between frames in the training and test sets. The first four methods deal only with frames from a particular session. In the first method which we will refer to as the frame split, frames were independently drawn into the training and test sets. Although the draws were independent, there is no restriction to prevent adjacent frames being split between the training and test sets. In the second method which we will refer to as the block split, blocks of frames were independently drawn into the training and test sets. Similarly, epochs [I am currently calling half-runs epochs but we likely will want to change this and introduce the concept in the stimulus design section.] and entire runs of frames were independently drawn into the training and test sets to produce the epoch split and run split respectively. For the fifth method, we split entire sessions between training and test sets to produce the session split. This method is the only one to employ frames from different sessions in the training and test sets. In existing literature, this method is essentially identical to measures of between session classifier accuracy. This final method produces an extremely large average temporal distance between data in the test and training sets that is on the order of several months for some subjects.

For the frame and block splits, the classifier performances were estimated using 10-fold cross-validation. That is, the dataset was randomly split into the training and test sets 10 times and a classifier is trained on each split. The classifier's performance is then estimated as the average of its performance across all 10 splits. For the epoch, run, and session splits, only 8, 4, and 2 unique splits are possible respectively due to the much smaller number of runs and sessions per subject. Therefore, only 8-fold, 4-fold, and 2-fold cross-validation was employed for estimating classifier performance on these splits.

The performance of each classifier was characterized by its micro-averaged F-measure [6]. The F-measure for a single class is described by the following equations:

$$\text{precision} = \frac{tp}{tp + fp} \tag{6}$$

$$\text{recall} = \frac{tp}{tp + fn} \tag{7}$$

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{8}$$

Where tp is the number of true positives, fp is the number of false positives, and fn is the number of false negatives. The F-measure is a more robust measure of the performance of a classifier than either precision or recall alone. For example, if the classifier labeled everything as positive then the recall would be perfect but the precision would be at chance levels. On the other hand, if the classifier only labeled examples it was highly confidant in as positive then precision would be high but recall would be low. The F-measure can be generalized for multiple classes by summing true positive, false positive, and false negative counts across

all classes.

$$\text{precision}_{avg} = \frac{\sum_i^M tp_i}{\sum_i^M (tp_i + fp_i)} \tag{9}$$

$$\text{recall}_{avg} = \frac{\sum_i^M tp_i}{\sum_i^M (tp_i + fn_i)} \tag{10}$$

Where M is the number of classes. The multi-class F-measure is then calculated as:

$$F_{avg} = 2 \cdot \frac{\text{precision}_{avg} \cdot \text{recall}_{avg}}{\text{precision}_{avg} + \text{recall}_{avg}} \tag{11}$$

This is known in the literature as the micro-averaged F-measure. It should be noted that in a symmetric multi-class scheme such as ours, the micro-averaged precision, recall, and F-measure will all be identical.

Another convenient tool for examining classifier performance is the confusion matrix. If $\mathbf{C}$ is a confusion matrix, then the value of C_{ij} is equal to the number examples of class i that were classified as class j. Therefore, values along the diagonal of a confusion matrix correspond to correct classifications while other values correspond to incorrect classifications. The confusion matrix also simplifies calculating precision and recall for each class. The value of C_{ii} divided by the sum of all values along row i is the recall of the i^{th} class. Similarly, the value of C_{jj} divided by the sum of all values along column j is the precision of the j^{th} class. Finally, the micro-averaged F-measure can be calculated by dividing the sum along the diagonal, or the trace, by the sum of the entire matrix.

2.5 Sensitivity Analysis

Non-linear multi-variate machine learning classifiers can tell us whether a group of voxels is discriminative with respect to the task being predicted. However, it is not obvious which voxels in a large group were actually important for determining that discrimination. This is important for localizing functions in the brain. One existing technique is to train machine learning classifiers on small localized areas in the brain and use their performance as a measure of the strength of the function in question in that area. While this technique is effective for simple highly localized functions, the results are less clear when the function is sparsely distributed over the brain. No one region may contain enough information for accurate predictions.

To overcome this limitation, we have trained our classifiers on large regions of the brain and used sensitivity analysis to attempt to tease out the sparsely distributed voxels that are relevant for task discrimination. Specifically, we calculate the sensitivity, or magnitude of change, of the output of the classifier with respect to a change in each voxel. In feed forward neural networks, this problem has been well explored [7]. Let $\mathbf{o}$ be the vector of outputs and $\mathbf{x}$ be the vector of inputs. Then the sensitivity of output k to input i is defined by:

$$S_{ki} = \frac{\delta o_k}{\delta x_i} \tag{12}$$

Or simply, the partial derivative of the output with respect to the input. If we let $\mathbf{w}$ be the weight matrix from the hidden layer to the output layer and $\mathbf{v}$ be the weight matrix from the input layer to the hidden layer then the partial derivative can be expressed as follows:

$$\frac{\delta o_k}{\delta x_i} = o'_k \sum_{j=1}^{J} w_{kj} y'_j v_{ji} \tag{13}$$

Where J is the total number of hidden neurons, o'_k is the value of the derivative of the activation function at output k, and y'_j is the value of the derivative of the activation function at hidden neuron j. Finally, the entire sensitivity matrix can be expressed in matrix notation as:

$$\mathbf{S} = \mathbf{O}' \times \mathbf{W} \times \mathbf{Y}' \times \mathbf{V} \tag{14}$$

Where

$$\mathbf{O}' = diag(o'_1,\ o'_2,\ \cdots,\ o'_K) \tag{15}$$

$$\mathbf{Y}' = diag(y'_1,\ y'_2,\ \cdots,\ y'_K) \tag{16}$$

However, because the transfer functions are non-linear they can only be evaluated for specific input values. Therefore, we calculate the average sensitivity matrix across all input vectors.

$$\mathbf{S}_{avg} = \sqrt{\frac{\sum_{n=1}^{N} (\mathbf{S}^n)^2}{N}} \tag{17}$$

Where N is the number of input vectors. The magnitude is squared to avoid problems with positive and negative sensitivities canceling out when averaging. The average of the absolute value of sensitivities could also be employed. This still gives a sensitivity value for each voxel with respect to every output, whereas it is useful to have a measure of the sensitivity of a voxel with respect to any output. To calculate this number we simply take the maximum sensitivity of each voxel across all outputs.

$$\Phi_i = \max_{k=1\ldots K} S_{ki,\ avg} \tag{18}$$

This sensitivity can now be projected back into the volume anatomy space to create an activation map. Further, this sensitivity can also be used to reduce the dimensionality of the machine learning classifier by retraining the algorithms using only the voxels that had a sensitivity over some threshold.

3 Results

4 Conclusions

[The conclusion is still mostly pulled from the HBM abstract. I would like to rewrite this after the rest of the paper is more complete. Especially with regards to future research directions.]

	Frame		Block		Epoch		Run		Session	
Subject	SVM	NN	SVM	NN	SVM	NN	SVM	NN	SVM	NN
A	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
B	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
C	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
D	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
E	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
Average	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6

Table 1: The multi-class F_1 scores of the linear SVM and the feed forward neural network after a 10 fold cross-validation for all 5 subjects. Every frame was shuffled independently into the train, test, and validation sets.

	Frame		Block		Epoch		Run		Session	
Subject	SVM	NN	SVM	NN	SVM	NN	SVM	NN	SVM	NN
A	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
B	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
C	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
D	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
E	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
Average	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6

Table 2: The multi-class F_1 scores of the linear SVM and the feed forward neural network after a 10 fold cross-validation for all 5 subjects. Every block was shuffled independently into the train, test, and vaidation sets.

	Frame		Block		Epoch		Run		Session	
Subject	SVM	NN	SVM	NN	SVM	NN	SVM	NN	SVM	NN
A	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
B	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
C	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
D	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
E	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
Average	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6

Table 3: The multi-class F_1 scores of the linear SVM and the feed forward neural network after an 8 fold cross-validation for all 5 subjects. Every epoch was shuffled independently into the train, test, and vaidation sets. Only 8 folds were used because most subjects have only 8 epochs of data collected.

	Frame		Block		Epoch		Run		Session	
Subject	SVM	NN	SVM	NN	SVM	NN	SVM	NN	SVM	NN
A	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
B	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
C	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
D	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
E	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
Average	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6

Table 4: The multi-class F_1 scores of the linear SVM and the feed forward neural network after a 4 fold cross-validation for all 5 subjects. Every run was shuffled independently into the train, test, and vaidation sets. Only 4 folds were used because most subjects have only 4 runs of data collected.

	Frame		Block		Epoch		Run		Session	
Subject	SVM	NN	SVM	NN	SVM	NN	SVM	NN	SVM	NN
A	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
B	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
C	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
D	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
E	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
Average	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6

Table 5: The multi-class F_1 scores of the linear SVM and the feed forward neural network after a 2 fold cross-validation for all 5 subjects. Every session was shuffled independently into the train, test, and vaidation sets. Only 2 folds were used because most subjects have only 2 sessions of data collected.

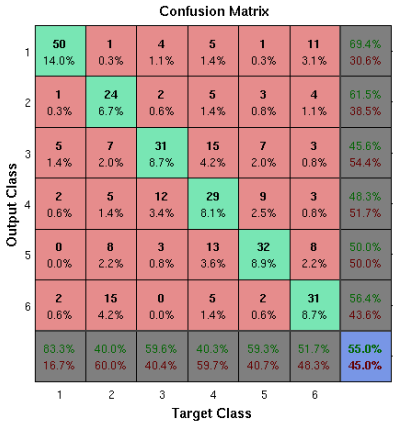

Figure 6: The average confusion matrix across all subjects when the train, test, and validation sets were divided by epoch.

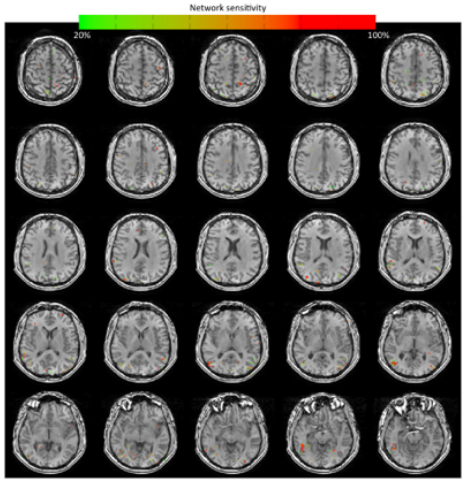

Figure 7: The results of the sensitivity analysis mapped back on to the volume anatomy.

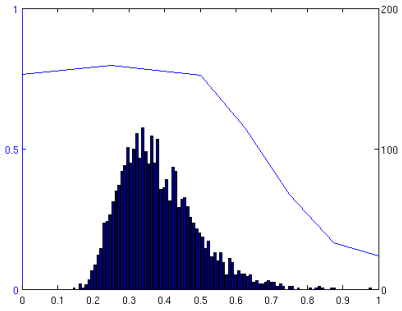

Figure 8: A histogram of the sensitivity analysis values and a plot of the feed forward neural network F_1 score when the inputs are pruned at a particular sensitivity value.

The reported results are well above chance, indicating there is useful information about character number in the BOLD signal. The sensitivity analysis indicates that no single region of the brain is responsible for counting characters, and there is not a simple linear relationship between magnitude of activation and cardinality. Rather, it is a complex pattern of distributed activation requiring machine learning methods to capture the stimulus-response relationship.

Earlier, we presented this new trend in brain sate classification as a departure from traditional fMRI experiments which seek to identify the purpose or function of particular brain regions. However, it is important to note that through sensitivity analysis these machine learning classifiers can be re-purposed for just that goal. If a region of the brain is highly important for accurately predicting the presence of a particular stimulus then it logically follows that that region must somehow be involved in the processing of that stimulus. Furthermore, multi-voxel non-linear machine learning classifiers can potentially identify much more complex interactions between brain regions than the simple GLM.

References

[1] Boynton, G. M., Engel, S. a., Glover, G. H., and Heeger, D. J. Linear systems analysis of functional magnetic resonance imaging in human V1. *The Journal of neuroscience : the official journal of the Society for Neuroscience 16*, 13 (July 1996), 4207–21.

[2] Haxby, J. V., Gobbini, M. I., Furey, M. L., Ishai, a., Schouten, J. L., and Pietrini, P. Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science (New York, N.Y.) 293*, 5539 (Sept. 2001), 2425–30.

[3] Haynes, J., and Rees, G. Decoding mental states from brain activity in humans. *Nature Reviews Neuroscience* (2006).

[4] Kohavi, R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *International Joint Conference on Artificial Intelligence* (1995), pp. 1137–1145.

[5] Mitchell, T., Hutchinson, R., Just, M., Niculescu, R., Pereira, F., and Wang, X. Classifying instantaneous cognitive states from fMRI data. In *AMIA Annual Symposium Proceedings* (2003), pp. 465–469.

[6] Özgür, A., Özgür, L., and Güngör, T. Text Categorization with Class-Based and Corpus-Based Keyword Selection. In *Computer and Information Sciences - ISCIS 2005*, P. Yolum, T. Güngör, F. Gürgen, and C. Özturan, Eds., vol. 3733 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 606–615.

[7] Zurada, J., Malinowski, A., and Cloete, I. Sensitivity analysis for minimization of input data dimension for feedforward neural network. In *Proceedings of IEEE International Symposium on Circuits and Systems - ISCAS '94* (1994), vol. 6, IEEE, pp. 447–450.