Classifying Structural MRI Scans of Patients with Major Depressive Disorder

Joe Marrama, Christie Paz, Guillaume Davo, Estelle Comment

***Abstract*—The goal of this paper is to determine if machine learning algorithms can be applied to distinguish differences between structural MRI brain scans of patients with Major Depressive Disorder and those of healthy patients. A large range of standard machine learning techniques were tested in this study. While a majority of these suffer from the small size of the dataset we used and thus are prone to overfitting, it appears that Deep Belief Nets can be trained for this classifying task with a reasonable accuracy.**

I. Problem Statement

Major Depressive Disorder (MDD) is a serious mental ailment that affects 10-25% of women and 5-12% of men. Symptoms include headaches, irritability, a feeling of worthlessness, and thoughts of suicide. Although 2/3 of people diagnosed with MDD at some point in their life recover completely, family members may be 1.5 to 3 times more likely to develop this condition.

Currently, the diagnosis of MDD relies upon self-reported experiences of a patient. These diagnoses are often biased. MDD patients may deny that their feelings are abnormal. For this reason, it is critical that unbiased tests be developed to diagnose this disorder.

Recent research using machine learning on medical images, such as fMRI and MRI scans, has proved to be successful for many diagnosis algorithms, such as drug addiction [1]; Ecker et. al. [2] recently used support vector machines (SVM) on MRI scans to diagnose autism with accuracies over 80%.

In this paper, we will discuss several algorithms we used to classify structural MR images of patients with MDD and controls. In the future, these algorithms may be used to develop a clinical diagnosis for MDD and may even provide insight on the causes of this disorder.

II. Materials

All participants were scanned at Stanford University using high resolution 1-Tesla or 3-T scanners. We acquired 76 brain scans (24 Female MDD, 11 Male MDD, 29 Female Controls, 12 Male Controls) from subjects of various ages (18-60). The scans were then normalized so that each of the 121x145x121 voxels in the image mapped to approximately the same point in 3D space within the brain.

Given the large number of features and small number of examples, our primary problem was, to reduce our feature space to avoid overfitting while minimizing the training and test errors. One solution was to reduce the resolution of our dataset by averaging. In this paper, we will refer to reduced resolution data sets as lo2 for resolution reduced by 2, lo3 for resolution reduced by 3, etc.

Most of our experiments were done on a balanced set of 68 examples. The training and test sets were then randomly selected (with a ratio of about 70% / 30%).

III. Classifiers

A. Support Vector Machine with L1 Regularization

In an SVM, L1-regularization encourages sparsity in the w coefficients. In our problem where we have too many features, we can use it as a method of feature selection. The L1 norm regularization is equivalent to :

$$\min_w \frac{1}{C}||w||_1 + \sum_{i=1}^{l} \max\left(0, 1 - y_i w^T x_i\right)^2.$$

The smaller C is, the more the coefficients of w are reduced.

The optimal value of C was found to be around 0.2, for which the mean test error was 41.9%, with a standard deviation of 12.1%.

The non-zero coefficients of w correspond to points in the 3D scan, that we can plot against the map of the brain (see figure 1). These are the features selected by the SVM.

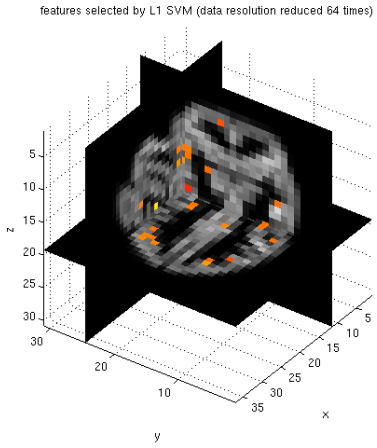

Figure 1. coefficents of features selected by L1 SVM, on lo4 data, with c=0.2

B. Naive Bayes

We initially tried Naïve Bayes as a benchmark algorithm. We discretized the data with equally spaced bins, and tested 50 times on different resolutions of data, taking out the background features. NB performed well compared to other methods: we know that in cases where there are few examples,

Table I
NAIVE BAYES TEST ERROR, MEAN +/- STD, ON ORIGINAL DATA

	10 bins	20 bins	30 bins
lo5	50.48 +/- 9.08	49.14 +/- 9.49	48.67 +/- 10.3
lo4	48.38 +/- 8.41	45.81 +/- 8.81	48.38 +/- 7.16
lo3	48.00 +/- 9.99	46.48 +/- 8.95	47.24 +/- 9.07
lo2	47.14 +/- 9.54	46.10 +/- 8.99	47.52 +/- 8.79
full resolution	46.19 +/- 11.19	**43.62 +/- 7.85**	46.48 +/ 9.55

it can outperform SVM. It seemed to perform best on the full set of features (see table I).

We also tried NB on gradient maps of the data, taking only the binned direction into account (8 and 32 bins of direction.). This also performed well on the full size data, but much worse (~50%) on the lower resolutions (see table II).

Table II
NAIVE BAYES TEST ERROR, MEAN +/- STD, ON GRADIENT DATA

	8 bins	32 bins
lo4	52.58 +/-12.07	52.95 +/- 8.64
lo2	44.48 +/- 8.83	46.48 +/- 9.97
full resolution	**43.43 +/- 9.89**	43.90 +/- 8.62

C. Logistic Regression

We implemented a logistic regression algorithm with various learning rates on the lo5 data set. Our algorithm iterated through each training sample (of 53 subjects, 70% of our total data) and each feature until the value of theta converged to a learning threshold of 0.015. We then classified the test set using the converged value of theta. Figure 2 summarizes our results. Unfortunately, this algorithm did not have significant test error. However, the training error was 0 for all iterations.

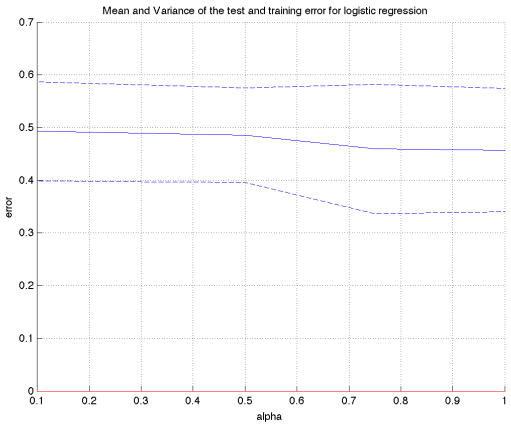

Figure 2. Average test error of logistic regression on lo5 data over 20 iterations, training over 70% data

D. Student's t-test

T-test Student's t-distribution is a continuous probability distribution that resembles a normal distribution where the sample size is small. T-testing uses student's t-distributions to determine if two sets of data are significantly different- i.e. if the null hypothesis can be rejected. Using AFNI's open source toolbox, we created a t-test mask from our two sets of data- MDDs versus controls. This mask contained coefficients for each voxel that were proportional to the likelihood that the difference between voxel values in these two sets were significant. (Voxel values that appeared to be significantly different in the two sets had higher coefficients.) Our raw data was then multiplied by this mask and put into an SVM. Using leave-one-out cross validation and reducing resolution of the new data set by a factor of 5, our average test error was 0.45. Using L1 normalization also did not prove to be effective; our test error was 0.56.

E. Handpicked Anatomical Feature Mask

Given previous psychological research on MDD, we reduced our feature space to include only those voxels that were in areas of the brain that were already thought to be related to MDD. This reduced our feature space to only 2263 features per sample. We then fed this in to our support vector machine using leave-one-out cross-validation to determine the error.

Unfortunately, this method failed to classify MRI images with reliable accuracy (See fig. 3). The test error of leave-one-out cross-validation was 0.505. Figure 3 also shows that the MDD test errors tended to be smaller than the test errors of control, but not significant enough to suggest that the SVM favored one classification over the other.

Since MDD affects twice as many women as it does men, we compared the test errors between females and males to see if there were any significant differences. In fig. 3, you can see the test errors of support vector machines that were trained on just female data or just male data. It's interesting to note that the test errors are not that different even though female are more likely to have MDD than males.

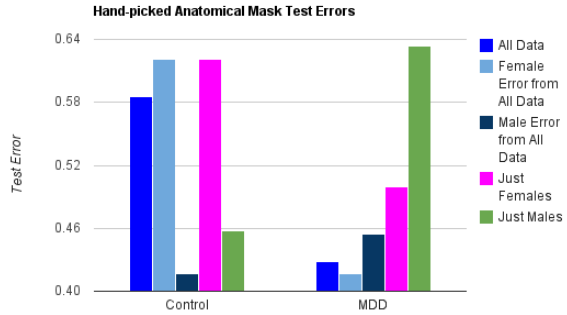

Figure 3. est errors from hand-picked anatomical feature mask from leave-one-out cross-validation of lo5 data of a sample set composed of all data, a sample set composed only of females and a sample set composed only of males

F. Principal Component Analysis

Principal Component Analysis is used to massively reduce the features space dimension. To test the efficiency of PCA on our problem, we used a balanced set of 68 examples. The

training and test sets were then randomly selected (with a ratio of about 70% / 30%). Since our data represents images, the mean (training) image was subtracted from the examples but no variance scaling was done. After projecting both the training and test examples on the principle components, we ran a soft-margin SVM with linear and Gaussian Radial Basis Function kernels. Classification results were quite unsatisfactory (see figure 4).

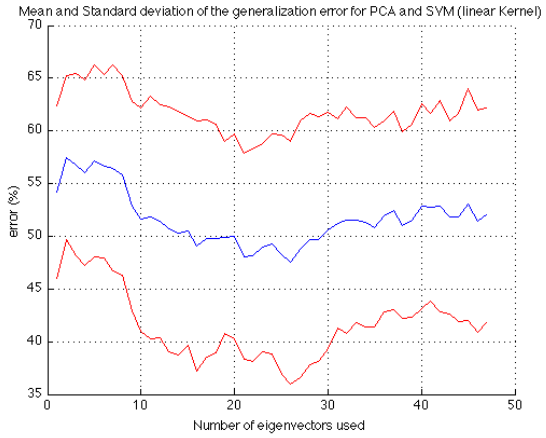

Figure 4. Mean (blue) ± standard deviation (red) on average for 50 trials, versus number K of eigenvectors used for SVM. The first K eigenvectors are used.

The values are averaged for 50 trials (random permutations of our original data set). The mean error is around 50% so it seems that there is no information left at all relative to our classification task after PCA. However it can be noted that the error is minimal when a median number of eigenvectors is used (about 25), as expected. Using more eigenvectors gives indeed more information on the initial features, but energy is concentrated on the first eigenvectors so using the last eigenvectors worsen performance. To understand why PCA performs so poorly, we studied the features projections on two of the main components. Unsurprisingly, the data does not look separable at all, hence the difficulty for SVM to classify on the test set.

G. Feature Selection by Mutual Information

A classic method for feature selection is to chose those who have the best mutual information with the labels:

$$I(X_i, Y) = \sum_{x,y} p(X_i = x, Y = y) \log \left(\frac{p(X_i = x, Y = y)}{p(X_i = x)p(Y = y)} \right)$$

The probabilities were estimated by maximum likelihood (by counting the occurrences and normalizing) and Laplace-smoothed. Mutual information failed because we had so few examples and so many features; in all the available features, it was always possible to find a feature that correlated perfectly with the labels, by chance. This was usually a noise feature that had escaped denoising, out of the brain.

H. Forward Feature Search

Since mutual information failed, we tried forward feature search to select the best features, using L1 SVM. Each feature's performance was cross validated in the training set to determine the best. This method didn't yield good results either, and seemed to select features rather uniformly across the brain (see figure 5). It also had a very long running time. On lo4, the average test error was 49.35%. with a standard error of 10.82%.

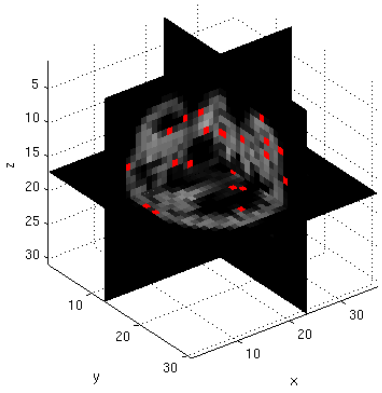

Figure 5. features selected by forward search, on lo4

I. Forward block feature search

We also derived an other algorithm form forward search for feature selection. Instead of selecting each feature corresponding to a voxel individually during the search, we divided the full resolution scans of the brain into 10x10x10 cubes and selected a subset of brain cubes for our features with forward search. Ten-fold cross-validation was used at each step to choose which cube to include in the feature set. We hoped this method could help us select specific brain areas related to MDD.

We also derived another algorithm from forward search for feature selection. Instead of selecting each feature corresponding to a voxel individually during the search, we divided the full resolution scans of the brain into 10x10x10 cubes and selected a subset of brain cubes for our features with forward search. Ten-fold cross-validation was used at each step to choose which cube to include in the feature set. We hoped this method could help us select specific brain areas related to MDD. However since the sparsity of the points giving the best results for the SVM is high, the cross-validation error profile is usually a step function and the test error is close to 50 % (see figure 6).

J. Histogram of Oriented Gradients

Most feature selection techniques and algorithms we used do not really take into account the very nature of the data, which is basically 3D images. On the contrary HOG descriptors are features descriptors used in image processing for object detection. So we tried to transform our dataset into HOG features and then run an SVM classifier. We derived a simplified version of the algorithm described in [3]. First the three components of the gradient (dx, dy, dz) are computed

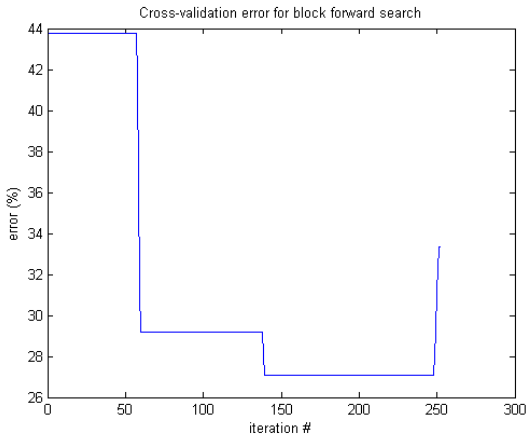

Figure 6. typical cross-validation error profile on the full resolution data set

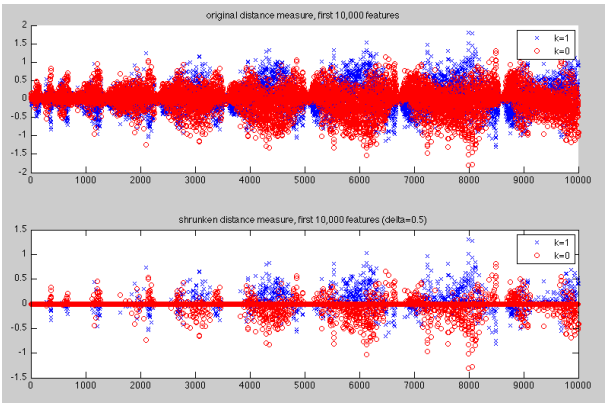

Figure 7. distance measures, original and shrunk The periodic variations correspond to sections through the brain.

for each image in the dataset. Then these components are used to bin the gradient direction. We tried to divide the space into 8 and 32 solid angles. After binning, the image is divided into 8x8x8 cells and a histogram of oriented gradients is built for each cell: each voxel votes for its bin and its vote is weighted with the norm of its gradient. Finally, histograms are juxtaposed to form the new features vector. Unfortunately, the SVM classification on these new features yields unsuccessful results. For 100 trials on the full resolution images, mean test error was 48.95% with a standard deviation of 8.32%, for 32 bins. Lower resolution data sets were also tested without success.

K. Nearest Shrunken Centroids

NSC is a regularized version of Gaussian Discriminant Analysis, with an assumption of independence between features. This assumption is reasonable since with our small number of examples, we could not have estimated the full covariance matrix anyway. To the standard gaussian analysis, we add a soft thresholding of the normalized distance between each feature's mean and class centroid [4]. Even though the shrinking seems to be active, as seen in figure the results are close to 50% test error.

NSC is a regularized version of Gaussian Discriminant Analysis, with an assumption of independence between features. This assumption is reasonable since with our small number of examples, we could not have estimated the full covariance matrix anyway.

To do the standard gaussian analysis, we add a soft thresholding of the normalized distance between each feature's mean and class centroid [?].

Even though the shrinking seems to be active, as seen in figure 7, the results are close to 50% test error.

L. Deep Belief Networks

To help reduce the high dimensionality of the data, we experimented with running the input through Deep Belief Nets and variants thereof in hopes of effectively preserving the information in the data while condensing it. Using a process similar to gradient ascent, we went through all of our preprocessing methods, post-processing classification techniques, and our variants on deep belief nets in order to find the best combination. Surprisingly, in the end, it was a single Restricted Boltzmann Machine with a SVM on top, trained on the non-compressed dataset, that performed the best. A discussion of our methods and results follow.

1) Preprocessing Methods : By just flattening the 121-by-145-by121 MRI scan matrices, the DBN variants we tried didn't get the best performance. Each example had 2122945 points, of which the majority were 0. Going through the complete data matrix (with each row as an example) and removing each column that contained all zeros, we reduced the dimension of each example to 493853. This resulting "noZeroColumn" dataset performed much better with all our models, because the columns that contained all 0's contain no helpful information, and likely correspond to points outside of the brain on the MRI scans.

As described before, we also tried crudely merging the points adjacent to each other in the MRI scan. Using both these compressed datasets and these datasets further processed with all zero columns removed didn't help performance in any of our DBN models.

2) Post-Processing Classification Methods : Without exception, the highest performing classification algorithm on the processed data from every DBN model was a L1 regularized SVM with a C value of 0.2. Naive Bayes typically performed the worst, and standard SVMs did slightly better. L2 regulated SVMs typically did much better than both Naive Bayes and normal SVMs, but the L1 SVM with a C value of 0.2 always did better than the rest, as the same on the normal dataset without DBN processing.

3) Deep Belief Net models and results : For all of our Deep Belief Net models, we used a standard RBM with a sparsity target for the activations of the hidden units [5, 6]. The sparsity target was .1, the learning rate for the sparsity was .03, and the learning rate for all other weight/bias updates was .01. We let the overall training loop run for 100 epochs, although the error usually converged well before 100 iterations.

The first model we tried was a standard RBM with varying numbers of hidden units. The standard RBM by far got the

best performance on the full "noZeroColumn" dataset. Unfortunately, due to the high dimension of the data, we could only use up to 750 hidden units on the corn clusters. The graphs below show the performance of these models. Surprisingly, the RBM performed much worse on all the condensed datasets, with and without the zero columns removed. Trained on each and every condensed dataset, the RBM got around 50% test error.(see figure 8).

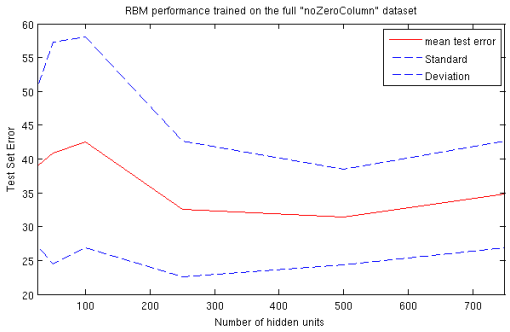

Figure 8. RBM test set performance as a function of the number of hidden units

Our next model was a standard 3-layer DBN. On the full "noZeroColumn" dataset, this process greatly worsened our performance from a single RBM. Because the first hidden layer is so much smaller then the dimension of the data, we suspect that the second and third layer can't extract any useful features from the first layer, rather they just "get in the way". We trained 3-layer DBNs on the compressed datasets (the smallest of our datasets had 4041 features, which allowed us to have comparable numbers of hidden units), but all DBNs trained on the compressed data still got an average of 50% test error.

We then tried training a RBM on only a small patch of the data, running all patches of that size through the RBM, and combining the resulting data in hopes of effectively reducing the feature size. For our first attempt at this, we used the "noZeroColumn" full dataset and sampled patches from the smoothed out data. Unfortuantely, for all patchsizes and factors we reduced the data's dimension by, our test error didn't deviate much from 50%. We also tried training a separate RBM on each patch, with the same results. However, the patches we sampled from in the "noZeroColumn" dataset correspond to generally uniformly distributed points in all areas of the brain, not localized patches. Thus, we next tried sampling from cubes in the brain instead of random patches and repeating the same methods (see figure 9) . This also didn't produce good results. For all brain cube sizes we tried, test error still didn't deviate far from 50%.

4) DBN Error Analysis : From both the performance of our models and the different datasets, a clear pattern emerges: all models that aren't trained on the full, non-compressed dataset end up performing poorly. Every run we did with a compressed dataset had test error close to 50%, as well as every model that trained an RBM or DBN (or multiple RBMs) on a subset of each example. Looking at larger patches from the MRI data of both normal and depressed brains, it becomes apparent

Figure 9. Top row: 15-by-15 voxel patches of the brain moving backwards; Bottom row: The resulting weights for these patches after being trained on the training set

that the data is very regular, and that there is no immediately discernible pattern usually found in one type of brain and not the other. Taking this into account, it makes sense that almost all of the techniques we tried to get around the high number of features failed on our MRI scans. This is because the indicative features of a depressed brain most likely lie in subtle points and relations between the points, which is easily lost in compression and convolution. Training an RBM on the full dataset must capture some of this information, which allows classification using SVMs to perform much better than without the RBM.

IV. Conclusions

Although many of the algorithms we implemented failed to be significantly accurate, deep belief networks proved to be the most promising. With more data from more subjects, our other algorithms would perform better with less variance. Although this problem was apparent when we first began this project, we had hoped to overcome this problem.

There are still many other algorithms that we can try, including LAR (Least Angle Regression) to find the best weight for L1 SVM, and recursive feature elimination. We can also explore more combinations of the various methods we tried.

References

[1] Ecker, Christine, et. al. "Investigating the predictive value of whole-brain structural MR scans in autism: A pattern classification approach." NeuroImage 49 (2010): 44-56.

[2] Zhang, Lei, et. al. "Machine learning for clinical diagnosis from functional magnetic resonance imaging." Computer Vision and Pattern Recognition, 2005.

[3] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: Computer Vision and Pattern Recognition, San Diego, CA, June 20–25, 2005.

[4] Trevor Hastie, Robert Tibshirani, Jerome Friedman, "The Elements of Statistical Learning: Data Mining, Inference, and Prediction", second edition, Feb. 2009, Springer.

[5] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. Neural Computation, 18(7):1527–1554, 2006.

[6] Chaitanya Ekanadham "Sparse deep belief net models for visual area V2". Stanford University. URL: http://ai.stanford.edu/~ang/papers/nips07-sparsedeepbeliefnetworkv2.ps